Bodystat SDK

Documentation for the Bodystat Software Developers Kit

Table of Contents

Bodystat API 1

Bodystat API - Main 1

C# .NET Wrapper 56

Bodystat API 57

Files 79

BodystatSDK.cs 80

Visual Basic .NET wrapper 80

Classes 81

BodystatAPI Class 81

BodystatAPI Structures 84

BodystatAPI Fields 114

BodystatAPI Methods 115

Structs, Records, Enums 145

BodystatAPI.BSAnalysisMode Enumeration 145

BodystatAPI.BSDeviceFamily Enumeration 146

BodystatAPI.BSDeviceModel Enumeration 147

BodystatAPI.BSError Enumeration 149

BodystatAPI.BSGender Enumeration 151

FAQs 152

Redistribution 154

Samples 155

Bodystat MFC Sample 155

Classes 155

CBodystatMFCDlg Class 156

CBodystatMFCDlg::CBodystatMFCDlg Constructor 158

CBodystatMFCDlg Data Members 159

CBodystatMFCDlg Methods 165

CBodystatResultsDlg Class 185

CBodystatResultsDlg::~CBodystatResultsDlg Destructor 186

CBodystatResultsDlg::CBodystatResultsDlg Constructor 186

CBodystatResultsDlg Data Members 187

CBodystatResultsDlg Methods 194

Bodystat C# WinForms Sample 204

BodystatCSWinForms Namespace 205

Classes 205

MainForm Class 205

ResultsForm Class 209

Files 215

MainForm.cs 216

ResultsForm.cs 216

Bodystat Visual Basic .NET Sample 216

BodystatVBNetWinFormSample Namespace 217

Classes 217

FrmBodystatMain Class 217

FrmBodystatResults Class 237

Index a

1 Bodystat API

The Bodystat API contains all the functions, definitions, classes and structures implemented by the Bodystat DLL.

The main API documentation is generic and is relevant to all programming languages. For your convenience, function definitions are provided in the documentation for popular languages (C++, C#, Java, Visual Basic & Matlab). However, you can call the API from any language capable of working with standard Windows DLLs based on C exports.

Language specific wrappers for the API are also provided for some languages. However, we regret we are unable to provide wrappers for all languages - we simply don't have the demand or in-house expertise to deal with them all! If you write a wrapper for a language we don't presently support directly we would be more than happy to include it in our SDK assuming you are willing to share your work with the community.

Modules

Name	**Description**
Bodystat API - Main (see page 1)	The main Bodystat API containing the functions, definitions, classes and structures implemented by the Bodystat DLL. The main API documentation is generic and is relevant to all programming languages. For your convenience, function definitions are provided in the documentation for popular languages (C++, C#, Java, Visual Basic & Matlab). However, you can call the API from any language capable of working with standard Windows DLLs based on C exports.
C# .NET Wrapper (see page 56)	Bodystat API C# .NET wrapper. Provides a wrapper interface to the Bodystat DLL with C# calls and types you can work with directly in your C# .NET program. The wrapper handles the necessary marshalling and interop issues required to interface with the native unmanaged Bodystat DLL.
Visual Basic .NET wrapper (see page 80)	Bodystat API Visual Basic .NET wrapper. Provides a wrapper interface to the Bodystat DLL with VB calls and types you can work with directly in your Visual Basic .NET program. The wrapper handles the necessary marshalling and interop issues required to interface with the native unmanaged Bodystat DLL.

1.1 Bodystat API - Main

The main Bodystat API containing the functions, definitions, classes and structures implemented by the Bodystat DLL.

The main API documentation is generic and is relevant to all programming languages. For your convenience, function definitions are provided in the documentation for popular languages (C++, C#, Java, Visual Basic & Matlab). However, you can call the API from any language capable of working with standard Windows DLLs based on C exports.

Namespaces

Name	**Description**
Bodystat (see page 2)	This is the main namespace for the Bodystat API. The entire functions within the Bodystat API are enclosed within the 'Bodystat' namespace.

Files

Name	Description
BodystatSDK.h (see page 55)	Bodystat SDK (C) 2010-2014 Bodystat (Isle of Man) Ltd. All rights reserved. Use subject to license. Web: http://www.bodystat.com Email: info@bodystat.com Tel: +44 (0)1624 629 571 Fax: +44 (0)1624 611 544 Version 1.36 (09/May/2014) Selected download is now passed through to the results form in the C# sample. Fixed error in VB.Net wrapper (preventing download working) in the VB.Net sample. Fixed some missing normals for MST device. 1.35 (10/Mar/2014) Added Cole methods for ECW/ICW calculations used in the MST device. Fixed a memory pointer issue for the detected bodystat device info structure. 1.34 (12/Nov/2013) Exposed Bluetooth API search timeout values to... more (see page 55)

Macros

Name	Description
BODYSTATSDK_API (see page 53)	The following ifdef block is the standard way of creating macros which make exporting from a DLL simpler. All files within this DLL are compiled with the BODYSTATSDK_EXPORTS symbol defined on the command line. This symbol should not be defined on any project that uses this DLL. This way any other project whose source files include this file see BODYSTATSDK_API functions as being imported from a DLL, whereas this DLL sees symbols defined with this macro as being exported.
M_PI (see page 54)	This is macro M_PI.

1.1.1 Bodystat API

This is the main namespace for the Bodystat API.

The entire functions within the Bodystat API are enclosed within the 'Bodystat' namespace.

Module

Bodystat API - Main (see page 1)

Functions

	Name	Description
◆	BSAuthenticateBTDevice (see page 8)	Authenticate (pair) a Bodystat device found during a previous call to BSSearchBTDevices (see page 22). Completely silent implementation.

◆	BSAutoSetupBT (see page 8)	Automatic Bluetooth setup. Automatically searches available Bluetooth devices, exchanges passkeys with Bodystat device and enables serial port service required for communication with the device. If multiple Bodystat devices are detected the user is prompted to chose the desired device (model and BT BDA address shown). This function is typically called as a one time setup for the device (e.g. after first installation, or from the hardware configuration page of your application's settings, or when the user obtains a replacement device (upgrades/replacement/etc). Typically, once setup you need only store the device com port returned. You can then subsequently communicate with the... more (see page 8)
◆	BSCalculateNormals (see page 10)	Calculate normal values and normal ranges for a given subject and their bio-impedance measurement together with the matching results structure for that measurement. You must supply a populated results structure which pertains to the measurement being supplied (obtained from BSCalculateResults (see page 10)).
◆	BSCalculateResults (see page 10)	Calculate complete result set for a given bio-impedance measurement recorded by the Bodystat device.
◆	BSCloseComport (see page 11)	Close com port.
◆	BSConnect (see page 11)	Connect to the Bodystat device and ensure the device is responding.
◆	BSGetBTBodystatDevice (see page 12)	Used to get information about any Bodystat device found after a previous call to BSSearchBTDevices (see page 22).
◆	BSGetBTStackInfo (see page 13)	Retrieve information about the Bluetooth stack. Useful for diagnostics.
◆	BSGetDeviceFamily (see page 13)	Determine the device family of a specific model.
◆	BSGetDeviceModelName (see page 14)	Determine the full product name of a specific model.
◆	BSGetDeviceModelNameShort (see page 14)	Determine the short product name of a specific model.
◆	BSGetSdkLibraryVersion (see page 15)	Determine the Bodystat SDK library (DLL) version.
◆	BSIsBTAvailable (see page 16)	Check if a supported BT radio is available on this computer. Checks for presence of USB Bluetooth dongles and inbuilt Bluetooth modules in the PC. No communication with Bodystat devices takes place during this query.
◆	BSOpenComport (see page 16)	Open com port to the device ready for communication.
◆	BSReadCalibrationTime (see page 17)	Read the calibration date from the connected device (when the unit was last calibrated). Only the date element of this value should be shown. Exact time value is not relevant.
◆	BSReadCurrentTime (see page 17)	Read the current time from the internal clock of the connected device. Note not all device models support this function (not all have an internal clock).
◆	BSReadModelVersion (see page 18)	Read device model and firmware version of the connected Bodystat device.
◆	BSReadPrinterAddress (see page 19)	Read the printer address the connected device is paired with (for direct printing). Not all device models support this function (not all support direct printing).
◆	BSReadProtocolInfo (see page 19)	Read protocol information from the connected device. Contains version numbers relating to communication protocols in use, data structures and auxiliary information.
◆	BSReadSerialNumber (see page 20)	Read the unique serial number of the connected Bodystat device.

	BSReadStoredTestData (see page 21)	Download test data from the connected device. Retrieves the last 100 test measurements (records) stored in the memory of the device. Downloading data does not affect the results stored in the memory - they remain unchanged. Note: you should store downloaded data locally if it is required permanently. Most devices store only the last 100 test records before overwriting the oldest records automatically.
	BSReadTestBodystat (see page 21)	Perform a quick bio-impedance test measurement on the connected bodystat device. Normal test process is skipped (user is not prompted to enter any subject information). Normal test protocol is disregarded (measurement proceeds regardless of lead status). Test measurement is not stored in device memory. May give wildly erratic readings if leads are not connected properly. Caller should sanity check results to detect for such scenarios. Results are intentionally not sanity checked to provide a true raw reading and provide caller with total discretion. Useful for diagnostics or performing measurements instigated by software.
	BSSearchBTDevices (see page 22)	Search BT radio for any Bodystat devices (paired or otherwise). Existing paired devices will be counted even if they are switched off or out of range. New non-paired units will only be counted if switched on, in range and ready for communication (on main startup screen).
	BSUnAuthenticateBTDevices (see page 23)	Unauthenticate (unpair) ALL Bodystat devices from this PC.
	BSWriteCurrentTime (see page 24)	Reset the current date/time of the internal clock on the connected device. Date/time is automatically set to match the current date/time of the PC
	BSWriteCurrentTimeAs (see page 24)	Reset the current date/time of the internal clock on the connected device. Date/time is set to a specific value of your choosing.
	BSWritePrinterAddress (see page 25)	Set the address of the printer the connected device should use for direct printing. Programmatic implementation of the inbuilt printer pairing function in the device.
	ESCalculateNormals (see page 26)	Calculate normal values and normal ranges for a given animal and their bio-impedance measurement together with the matching results structure for that measurement. You must supply a populated results structure which pertains to the measurement being supplied (obtained from ESCalculateResults (see page 26)).
	ESCalculateResults (see page 26)	Calculate complete result set for a given bio-impedance measurement recorded by the Equistat device.
	ESReadStoredTestData (see page 27)	Download test data from the connected device. Retrieves the last 100 test measurements (records) stored in the memory of the device. Downloading data does not affect the results stored in the memory - they remain unchanged. Note: you should store downloaded data locally if it is required permanently. Most devices store only the last 100 test records before overwriting the oldest records automatically.
	ESReadTestEquistat (see page 28)	Perform a quick bio-impedance test measurement on the connected equistat device. Normal test process is skipped (user is not prompted to enter any horse information). Normal test protocol is disregarded (measurement proceeds regardless of lead status). Test measurement is not stored in device memory. May give wildly erratic readings if leads are not connected properly. Caller should sanity check results to detect for such scenarios. Results are intentionally not sanity checked to provide a true raw reading and provide caller with total discretion. Useful for diagnostics or performing measurements instigated by software.

Structs, Records, Enums

	Name	Description

▣	BSAnalysisMode (see page 30)	BSAnalysisMode: Analysis results are available in several formats:
		BSAnalysisModeBC= Body composition analysis
		BSAnalysisModeHN= Hydration analysis
		BSAnalysisModeBoth= Combined body composition and hydration analysis (default)

▣	BSDeviceFamily (see page 30)	BSDeviceFamily: Enumerated values that group each device / model variant into a given product family. Devices within a given product family share broad capabilities and features. For details regarding capabilities and features refer to the technical documentation for that product family (http://www.bodystat.com)
▣	BSDeviceModel (see page 32)	BSDeviceModel: Enumerated values that represent the various different models of the Bodystat devices over the years. Suffix: The suffix after the model indicates specific hardware variations regarding connectivity: OPTO indicates model communicates via opto-isolated serial. DIU indicates model communicates via a data interface unit (induction interface) BT indicates model communicates via Bluetooth serial port.
▣	BSError (see page 33)	BSError: Bodystat Error codes. A list of Bodystat specific error codes.
▣	BSGender (see page 35)	BSGender: Enumerated values that represent a person's gender.
◆	BSMeasurement (see page 36)	BSMeasurement: Measurement structure. Each test measurement recorded in the device is stored one of these structures. Holds the user inputted information about the subject, together with bio-impedance readings measured by the device (raw electrical readings only, not full results).
◆	BSNormals (see page 38)	BSNormals: Bodystat Normals structure. Holds the normal values or normal ranges for any given test measurement.
◆	BSRawData (see page 40)	BSRawData: Bodystat Raw Data structure. Used when downloading test measurements from the device. Contains a fixed array of measurement records together with information about the number of records occupied within the array and test record starting number.
◆	BSResults (see page 41)	BSResults: Bodystat Results structure. Holds the calculated results for any given test measurement.
◆	ESMeasurement (see page 44)	ESMeasurement: Equistat Measurement structure. Each test measurement recorded in the device is stored one of these structures. Holds the user inputted information about the horse, together with bio-impedance readings measured by the device (raw electrical readings only, not full results).
◆	ESNormals (see page 45)	ESNormals: Equistat Normals structure. Holds the normal values or normal ranges for any given test measurement.
◆	ESRawData (see page 47)	ESRawData: Equistat Raw Data structure. Used when downloading test measurements from the device. Contains a fixed array of measurement records together with information about the number of records occupied within the array and test record starting number.
◆	ESResults (see page 48)	ESResults: Equistat Results structure. Holds the calculated results for any given test measurement.
◆	ImpData (see page 49)	Represents one frequency measurement

Variables

Name	Description
BS_RAWDATA_ARRAYSIZE (see page 52)	Number of measurement records BSRawData (see page 40) structure should reserved space for (currently fixed)
ES_RAWDATA_ARRAYSIZE (see page 52)	Number of measurement records ESRawData (see page 47) structure should reserved space for (currently fixed)

1.1.1.1 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
◆	BSAuthenticateBTDevice (see page 8)	Authenticate (pair) a Bodystat device found during a previous call to BSSearchBTDevices (see page 22). Completely silent implementation.
◆	BSAutoSetupBT (see page 8)	Automatic Bluetooth setup. Automatically searches available Bluetooth devices, exchanges passkeys with Bodystat device and enables serial port service required for communication with the device. If multiple Bodystat devices are detected the user is prompted to chose the desired device (model and BT BDA address shown). This function is typically called as a one time setup for the device (e.g. after first installation, or from the hardware configuration page of your application's settings, or when the user obtains a replacement device (upgrades/replacement/etc). Typically, once setup you need only store the device com port returned. You can then subsequently communicate with the... more (see page 8)
◆	BSCalculateNormals (see page 10)	Calculate normal values and normal ranges for a given subject and their bio-impedance measurement together with the matching results structure for that measurement. You must supply a populated results structure which pertains to the measurement being supplied (obtained from BSCalculateResults (see page 10)).
◆	BSCalculateResults (see page 10)	Calculate complete result set for a given bio-impedance measurement recorded by the Bodystat device.
◆	BSCloseComport (see page 11)	Close com port.
◆	BSConnect (see page 11)	Connect to the Bodystat device and ensure the device is responding.
◆	BSGetBTBodystatDevice (see page 12)	Used to get information about any Bodystat device found after a previous call to BSSearchBTDevices (see page 22).
◆	BSGetBTStackInfo (see page 13)	Retrieve information about the Bluetooth stack. Useful for diagnostics.
◆	BSGetDeviceFamily (see page 13)	Determine the device family of a specific model.
◆	BSGetDeviceModelName (see page 14)	Determine the full product name of a specific model.
◆	BSGetDeviceModelNameShort (see page 14)	Determine the short product name of a specific model.
◆	BSGetSdkLibraryVersion (see page 15)	Determine the Bodystat SDK library (DLL) version.
◆	BSIsBTAvailable (see page 16)	Check if a supported BT radio is available on this computer. Checks for presence of USB Bluetooth dongles and inbuilt Bluetooth modules in the PC. No communication with Bodystat devices takes place during this query.
◆	BSOpenComport (see page 16)	Open com port to the device ready for communication.
◆	BSReadCalibrationTime (see page 17)	Read the calibration date from the connected device (when the unit was last calibrated). Only the date element of this value should be shown. Exact time value is not relevant.
◆	BSReadCurrentTime (see page 17)	Read the current time from the internal clock of the connected device. Note not all device models support this function (not all have an internal clock).
◆	BSReadModelVersion (see page 18)	Read device model and firmware version of the connected Bodystat device.
◆	BSReadPrinterAddress (see page 19)	Read the printer address the connected device is paired with (for direct printing). Not all device models support this function (not all support direct printing).

	BSReadProtocolInfo (see page 19)	Read protocol information from the connected device. Contains version numbers relating to communication protocols in use, data structures and auxiliary information.
	BSReadSerialNumber (see page 20)	Read the unique serial number of the connected Bodystat device.
	BSReadStoredTestData (see page 21)	Download test data from the connected device. Retrieves the last 100 test measurements (records) stored in the memory of the device. Downloading data does not affect the results stored in the memory - they remain unchanged. Note: you should store downloaded data locally if it is required permanently. Most devices store only the last 100 test records before overwriting the oldest records automatically.
	BSReadTestBodystat (see page 21)	Perform a quick bio-impedance test measurement on the connected bodystat device. Normal test process is skipped (user is not prompted to enter any subject information). Normal test protocol is disregarded (measurement proceeds regardless of lead status). Test measurement is not stored in device memory. May give wildly erratic readings if leads are not connected properly. Caller should sanity check results to detect for such scenarios. Results are intentionally not sanity checked to provide a true raw reading and provide caller with total discretion. Useful for diagnostics or performing measurements instigated by software.
	BSSearchBTDevices (see page 22)	Search BT radio for any Bodystat devices (paired or otherwise). Existing paired devices will be counted even if they are switched off or out of range. New non-paired units will only be counted if switched on, in range and ready for communication (on main startup screen).
	BSUnAuthenticateBTDevices (see page 23)	Unauthenticate (unpair) ALL Bodystat devices from this PC.
	BSWriteCurrentTime (see page 24)	Reset the current date/time of the internal clock on the connected device. Date/time is automatically set to match the current date/time of the PC
	BSWriteCurrentTimeAs (see page 24)	Reset the current date/time of the internal clock on the connected device. Date/time is set to a specific value of your choosing.
	BSWritePrinterAddress (see page 25)	Set the address of the printer the connected device should use for direct printing. Programmatic implementation of the inbuilt printer pairing function in the device.
	ESCalculateNormals (see page 26)	Calculate normal values and normal ranges for a given animal and their bio-impedance measurement together with the matching results structure for that measurement. You must supply a populated results structure which pertains to the measurement being supplied (obtained from ESCalculateResults (see page 26)).
	ESCalculateResults (see page 26)	Calculate complete result set for a given bio-impedance measurement recorded by the Equistat device.
	ESReadStoredTestData (see page 27)	Download test data from the connected device. Retrieves the last 100 test measurements (records) stored in the memory of the device. Downloading data does not affect the results stored in the memory - they remain unchanged. Note: you should store downloaded data locally if it is required permanently. Most devices store only the last 100 test records before overwriting the oldest records automatically.

	ESReadTestEquistat (see page 28)	Perform a quick bio-impedance test measurement on the connected equistat device. Normal test process is skipped (user is not prompted to enter any horse information). Normal test protocol is disregarded (measurement proceeds regardless of lead status). Test measurement is not stored in device memory. May give wildly erratic readings if leads are not connected properly. Caller should sanity check results to detect for such scenarios. Results are intentionally not sanity checked to provide a true raw reading and provide caller with total discretion. Useful for diagnostics or performing measurements instigated by software.

1.1.1.1.1 Bodystat::BSAuthenticateBTDevice Function

Authenticate (pair) a Bodystat device found during a previous call to BSSearchBTDevices (see page 22). Completely silent implementation.

C++

```
BODYSTATSDK_API BOOL BSAuthenticateBTDevice(HWND hWndParent = NULL, unsigned short iTimeout
= 7);
```

C#

```
BODYSTATSDK_API BOOL BSAuthenticateBTDevice(HWND hWndParent, unsigned short iTimeout);
```

Visual Basic

```
Function BSAuthenticateBTDevice(hWndParent As HWND = NULL, iTimeout As unsigned short = 7)
As BODYSTATSDK_API BOOL
```

Java

```
BSAuthenticateBTDevice
```

MATLAB

```
function [BODYSTATSDK_API BOOL] = BSAuthenticateBTDevice(, )
```

File

BodystatSDK.h (see page 55)

Parameters

Parameters	Description
HWND hWndParent = NULL	Handle of the parent window.
unsigned short iTimeout = 7	Timeout multiplier supplied to the Bluetooth radio search API. Controls the amount of time to spend searching for the device. This function will block for this duration so is better called from a thread. Each multiplier increment equates to about 1.28 seconds. Thus a timeout value of 10 = 12.8s. Maximum time permitted by the API is a multiplier of 45 (just under 1 minute), values higher than this will be capped at 45. Default value is 7 (about 10 seconds).

Returns

TRUE if it succeeds, FALSE if it fails.

Remarks

Note this process can be time consuming to complete and executes synchronously. Either call from a threaded function or illustrate operation in progress to the user e.g. by means of an indefinite progress bar or busy mouse cursor, until the task completes or timesout.

1.1.1.1.2 Bodystat::BSAutoSetupBT Function

Automatic Bluetooth setup. Automatically searches available Bluetooth devices, exchanges passkeys with Bodystat device and enables serial port service required for communication with the device. If multiple Bodystat devices are detected the user is prompted to chose the desired device (model and BT BDA address shown).

This function is typically called as a one time setup for the device (e.g. after first installation, or from the hardware configuration page of your application's settings, or when the user obtains a replacement device (upgrades/replacement/etc).

Typically, once setup you need only store the device com port returned. You can then subsequently communicate with the device knowing only the com port. It is not necessary to setup the device in this way before each use.

New non-paired units will only be found if switched on, in range and ready for communication (on main startup screen). It is advisable to instruct the user to ensure this is the case before calling.

C++

```
BODYSTATSDK_API BOOL BSAutoSetupBT(LPTSTR lpComPort, int iSize, BOOL bReportErrors, HWND
hWndParent = NULL);
```

C#

```
BODYSTATSDK_API BOOL BSAutoSetupBT(LPTSTR lpComPort, int iSize, BOOL bReportErrors, HWND
hWndParent);
```

Visual Basic

```
Function BSAutoSetupBT(lpComPort As LPTSTR, iSize As Integer, bReportErrors As BOOL,
hWndParent As HWND = NULL) As BODYSTATSDK_API BOOL
```

Java

```
BSAutoSetupBT
```

MATLAB

```
function [BODYSTATSDK_API BOOL] = BSAutoSetupBT(, , , )
```

File

BodystatSDK.h (see page 55)

Parameters

Parameters	Description
LPTSTR lpComPort	[out] String buffer to receive the device port the device is assigned to.
int iSize	The size of the buffer.
BOOL bReportErrors	TRUE to report errors to the user (shown as message boxes, locale constraints apply). FALSE to hide errors from the user (still not completely silent).
HWND hWndParent = NULL	Handle of the parent window.

Returns

TRUE if it succeeds, FALSE if it fails.

Remarks

Note this process can be time consuming to complete and executes synchronously. Either call from a threaded function or illustrate operation in progress to the user e.g. by means of an indefinite progress bar or busy mouse cursor, until the task completes or timesout.

Errors during the process can be optionally shown/hidden. However, a completely silent approach is not possible using this function (instead use BSSearchBTDevices (see page 22) and BSGetBTBodystatDevice (see page 12) or iterate radio handles manually. Then pass desired handle to BSAuthenticateBTDevice (see page 8) for pairing).

Finally be aware this function is only available for supported locales. Otherwise English messages may be displayed to the user. Adopt silent approach discussed previously if you are targeting a non-supported locale.

1.1.1.1.3 Bodystat::BSCalculateNormals Function

Calculate normal values and normal ranges for a given subject and their bio-impedance measurement together with the matching results structure for that measurement. You must supply a populated results structure which pertains to the measurement being supplied (obtained from BSCalculateResults (see page 10)).

C++

```
BODYSTATSDK_API BSError BSCalculateNormals(const BSMeasurement * pM, const BSResults * pR,
BSNormals * pN);
```

C#

```
BODYSTATSDK_API BSError BSCalculateNormals(BSMeasurement * pM, BSResults * pR, ref
BSNormals pN);
```

Visual Basic

```
Function BSCalculateNormals(pM As BSMeasurement *, pR As BSResults *, ByRef pN As
BSNormals) As BODYSTATSDK_API BSError
```

Java

```
BSCalculateNormals
```

MATLAB

```
function [BODYSTATSDK_API BSError] = BSCalculateNormals(, , )
```

File

BodystatSDK.h (see page 55)

Parameters

Parameters	Description
const BSMeasurement * pM	The measurement details downloaded from the device (includes model, user inputted parameters and bio-impedance measurement readings).
const BSResults * pR	The results calculated for this measurement previously.
BSNormals * pN	[out] Receives the normal values and normal ranges for this measurement.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

Normals vary by subject attributes, measured bio-impedance values and calculated results.

1.1.1.1.4 Bodystat::BSCalculateResults Function

Calculate complete result set for a given bio-impedance measurement recorded by the Bodystat device.

C++

```
BODYSTATSDK_API BSError BSCalculateResults(const BSMeasurement * pM, BSResults * pR,
BSAnalysisMode iAnalysisMode);
```

C#

```
BODYSTATSDK_API BSError BSCalculateResults(BSMeasurement * pM, ref BSResults pR,
BSAnalysisMode iAnalysisMode);
```

Visual Basic

```
Function BSCalculateResults(pM As BSMeasurement *, ByRef pR As BSResults, iAnalysisMode As
BSAnalysisMode) As BODYSTATSDK_API BSError
```

Java

```
BSCalculateResults
```

MATLAB

```
function [BODYSTATSDK_API BSError] = BSCalculateResults(, , )
```

File

BodystatSDK.h (see page 55)

Parameters

Parameters	Description
const BSMeasurement * pM	The measurement details downloaded from the device (includes model, user inputted parameters and bio-impedance measurement readings).
BSResults * pR	[out] Receives the complete calculated results for this measurement.
BSAnalysisMode iAnalysisMode	The desired analysis mode (see BSAnalysisMode (see page 30) for more information).

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

Results vary by subject attributes and measured bio-impedance values.

1.1.1.1.5 Bodystat::BSCloseComport Function

Close com port.

C++

```
BODYSTATSDK_API void BSCloseComport();
```

C#

```
BODYSTATSDK_API void BSCloseComport();
```

Visual Basic

```
Function BSCloseComport() As BODYSTATSDK_API void
```

Java

```
BSCloseComport
```

MATLAB

```
function [BODYSTATSDK_API void] = BSCloseComport()
```

File

BodystatSDK.h (see page 55)

1.1.1.1.6 Bodystat::BSConnect Function

Connect to the Bodystat device and ensure the device is responding.

C++

```
BODYSTATSDK_API BSError BSConnect();
```

C#

```
BODYSTATSDK_API BSError BSConnect();
```

Visual Basic

```
Function BSConnect() As BODYSTATSDK_API BSError
```

Java

```
BSConnect
```

MATLAB

```
function [BODYSTATSDK_API BSError] = BSConnect()
```

File

BodystatSDK.h (see page 55)

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have opened the com port by calling BSOpenComPort previously.

1.1.1.1.7 Bodystat::BSGetBTBodystatDevice Function

Used to get information about any Bodystat device found after a previous call to BSSearchBTDevices (see page 22).

C++

```
BODYSTATSDK_API BOOL BSGetBTBodystatDevice(LPTSTR lpDeviceName, int iDeviceNameBufferSize,
LPTSTR lpBDA, int iBDABufferSize, LPTSTR lpComPort, int iComBufferSize, unsigned short
iTimeout = 7);
```

C#

```
BODYSTATSDK_API BOOL BSGetBTBodystatDevice(LPTSTR lpDeviceName, int iDeviceNameBufferSize,
LPTSTR lpBDA, int iBDABufferSize, LPTSTR lpComPort, int iComBufferSize, unsigned short
iTimeout);
```

Visual Basic

```
Function BSGetBTBodystatDevice(lpDeviceName As LPTSTR, iDeviceNameBufferSize As Integer,
lpBDA As LPTSTR, iBDABufferSize As Integer, lpComPort As LPTSTR, iComBufferSize As Integer,
iTimeout As unsigned short = 7) As BODYSTATSDK_API BOOL
```

Java

```
BSGetBTBodystatDevice
```

MATLAB

```
function [BODYSTATSDK_API BOOL] = BSGetBTBodystatDevice(, , , , , , )
```

File

BodystatSDK.h (see page 55)

Parameters

Parameters	Description
LPTSTR lpDeviceName	[out] String buffer to receive the product name of the Bodystat device.
int iDeviceNameBufferSize	Size of the device name buffer.
LPTSTR lpBDA	[out] String buffer to receive the Bluetooth hexadecimal address (known as the BDA - unique to each BT module in a device, like a MAC address).
int iBDABufferSize	Size of the BDA buffer.
LPTSTR lpComPort	[out] String buffer to receive device/port the unit is assigned to.
int iComBufferSize	Size of the com port buffer.

unsigned short iTimeout = 7	Timeout multiplier supplied to the Bluetooth radio search API. Controls the amount of time to spend searching for the device. This function will block for this duration so is better called from a thread. Each multiplier increment equates to about 1.28 seconds. Thus a timeout value of 10 = 12.8s. Maximum time permitted by the API is a multiplier of 45 (just under 1 minute), values higher than this will be capped at 45. Default value is 7 (about 10 seconds).

Returns

TRUE if it succeeds, FALSE if it fails.

1.1.1.1.8 Bodystat::BSGetBTStackInfo Function

Retrieve information about the Bluetooth stack. Useful for diagnostics.

C++

```
BODYSTATSDK_API BOOL BSGetBTStackInfo(LPTSTR lpBuffer, int iBufferSize);
```

C#

```
BODYSTATSDK_API BOOL BSGetBTStackInfo(LPTSTR lpBuffer, int iBufferSize);
```

Visual Basic

```
Function BSGetBTStackInfo(lpBuffer As LPTSTR, iBufferSize As Integer) As BODYSTATSDK_API
BOOL
```

Java

```
BSGetBTStackInfo
```

MATLAB

```
function [BODYSTATSDK_API BOOL] = BSGetBTStackInfo(, )
```

File

BodystatSDK.h (see page 55)

Parameters

Parameters	Description
LPTSTR lpBuffer	[out] String buffer to receive the information.
int iBufferSize	Size of the buffer.

Returns

true if it succeeds, false if it fails.

Remarks

Note: Information can only be provided for supported BT radios.

1.1.1.1.9 Bodystat::BSGetDeviceFamily Function

Determine the device family of a specific model.

C++

```
BODYSTATSDK_API BSDeviceFamily BSGetDeviceFamily(BSDeviceModel iModel);
```

C#

```
BODYSTATSDK_API BSDeviceFamily BSGetDeviceFamily(BSDeviceModel iModel);
```

Visual Basic

```
Function BSGetDeviceFamily(iModel As BSDeviceModel) As BODYSTATSDK_API BSDeviceFamily
```

Java

```
BSGetDeviceFamily
```

MATLAB

```
function [BODYSTATSDK_API BSDeviceFamily] = BSGetDeviceFamily()
```

File

BodystatSDK.h (see page 55)

Parameters

Parameters	Description
BSDeviceModel iModel	The desired model of interest.

Returns

The device family the model belongs to.

1.1.1.1.10 Bodystat::BSGetDeviceModelName Function

Determine the full product name of a specific model.

C++

```
BODYSTATSDK_API BSError BSGetDeviceModelName(BSDeviceModel iModel, LPTSTR lpBuffer, int
iBufferSize);
```

C#

```
BODYSTATSDK_API BSError BSGetDeviceModelName(BSDeviceModel iModel, LPTSTR lpBuffer, int
iBufferSize);
```

Visual Basic

```
Function BSGetDeviceModelName(iModel As BSDeviceModel, lpBuffer As LPTSTR, iBufferSize As
Integer) As BODYSTATSDK_API BSError
```

Java

```
BSGetDeviceModelName
```

MATLAB

```
function [BODYSTATSDK_API BSError] = BSGetDeviceModelName(, , )
```

File

BodystatSDK.h (see page 55)

Parameters

Parameters	Description
BSDeviceModel iModel	The desired model of interest.
LPTSTR lpBuffer	[out] String buffer to receive the name.
int iBufferSize	Size of the buffer.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

1.1.1.1.11 Bodystat::BSGetDeviceModelNameShort Function

Determine the short product name of a specific model.

C++

```
BODYSTATSDK_API BSError BSGetDeviceModelNameShort(BSDeviceModel iModel, LPTSTR lpBuffer,
int iBufferSize);
```

C#

```
BODYSTATSDK_API BSError BSGetDeviceModelNameShort(BSDeviceModel iModel, LPTSTR lpBuffer,
int iBufferSize);
```

Visual Basic

```
Function BSGetDeviceModelNameShort(iModel As BSDeviceModel, lpBuffer As LPTSTR, iBufferSize
As Integer) As BODYSTATSDK_API BSError
```

Java

```
BSGetDeviceModelNameShort
```

MATLAB

```
function [BODYSTATSDK_API BSError] = BSGetDeviceModelNameShort(, , )
```

File

BodystatSDK.h (see page 55)

Parameters

Parameters	Description
BSDeviceModel iModel	The desired model of interest.
LPTSTR lpBuffer	[out] String buffer to receive the name.
int iBufferSize	Size of the buffer.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

1.1.1.1.12 Bodystat::BSGetSdkLibraryVersion Function

Determine the Bodystat SDK library (DLL) version.

C++

```
BODYSTATSDK_API BSError BSGetSdkLibraryVersion(DWORD * pMajorSdkVer, DWORD * pMinorSdkVer);
```

C#

```
BODYSTATSDK_API BSError BSGetSdkLibraryVersion(ref DWORD pMajorSdkVer, ref DWORD
pMinorSdkVer);
```

Visual Basic

```
Function BSGetSdkLibraryVersion(ByRef pMajorSdkVer As DWORD, ByRef pMinorSdkVer As DWORD)
As BODYSTATSDK_API BSError
```

Java

```
BSGetSdkLibraryVersion
```

MATLAB

```
function [BODYSTATSDK_API BSError] = BSGetSdkLibraryVersion(, )
```

File

BodystatSDK.h (see page 55)

Parameters

Parameters	Description
DWORD * pMajorSdkVer	[out] Receives the major version number of the Bodystat SDK DLL.
DWORD * pMinorSdkVer	[out] Receives the minor version number of the Bodystat SDK DLL.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

Caller should check the version number of the DLL if utilising functionality noted only as being available in later revisions of the SDK. Or, alternatively, ensure the SDK DLL is updated appropriately during installation of your application.

1.1.1.1.13 Bodystat::BSIsBTAvailable Function

Check if a supported BT radio is available on this computer. Checks for presence of USB Bluetooth dongles and inbuilt Bluetooth modules in the PC. No communication with Bodystat devices takes place during this query.

C++

```
BODYSTATSDK_API BOOL BSIsBTAvailable();
```

C#

```
BODYSTATSDK_API BOOL BSIsBTAvailable();
```

Visual Basic

```
Function BSIsBTAvailable() As BODYSTATSDK_API BOOL
```

Java

```
BSIsBTAvailable
```

MATLAB

```
function [BODYSTATSDK_API BOOL] = BSIsBTAvailable()
```

File

BodystatSDK.h (see page 55)

Returns

Returns true if supported radio is available and enabled.

Remarks

Note: Only supported BT radios can be queried (presently those compatible with the Microsoft Bluetooth APIs and some Widdcomm models).

Also note, Bluetooth must be enabled on the PC by the user to permit detection. Some PCs, laptops particularly, have physical switches to enable bluetooth/wireless communication.

1.1.1.1.14 Bodystat::BSOpenComport Function

Open com port to the device ready for communication.

C++

```
BODYSTATSDK_API BSError BSOpenComport(LPCTSTR lpComPort, int iSize);
```

C#

```
BODYSTATSDK_API BSError BSOpenComport(LPCTSTR lpComPort, int iSize);
```

Visual Basic

```
Function BSOpenComport(lpComPort As LPCTSTR, iSize As Integer) As BODYSTATSDK_API BSError
```

Java

```
BSOpenComport
```

MATLAB

```
function [BODYSTATSDK_API BSError] = BSOpenComport(, )
```

File

BodystatSDK.h (see page 55)

Parameters

Parameters	Description
LPCTSTR lpComPort	String buffer containing the device/port to be opened.
int iSize	The size of the buffer.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

1.1.1.1.15 Bodystat::BSReadCalibrationTime Function

Read the calibration date from the connected device (when the unit was last calibrated). Only the date element of this value should be shown. Exact time value is not relevant.

C++

```
BODYSTATSDK_API BSError BSReadCalibrationTime(__time64_t * pCalibrationTime);
```

C#

```
BODYSTATSDK_API BSError BSReadCalibrationTime(ref __time64_t pCalibrationTime);
```

Visual Basic

```
Function BSReadCalibrationTime(ByRef pCalibrationTime As __time64_t) As BODYSTATSDK_API
BSError
```

Java

```
BSReadCalibrationTime
```

MATLAB

```
function [BODYSTATSDK_API BSError] = BSReadCalibrationTime()
```

File

BodystatSDK.h (see page 55)

Parameters

Parameters	Description
__time64_t * pCalibrationTime	[out] Receives the date the device was last calibrated.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect (see page 11) previously.

1.1.1.1.16 Bodystat::BSReadCurrentTime Function

Read the current time from the internal clock of the connected device. Note not all device models support this function (not all have an internal clock).

C++

```
BODYSTATSDK_API BSError BSReadCurrentTime(__time64_t * pCurrentTime, int iDST = -1);
```

C#

```
BODYSTATSDK_API BSError BSReadCurrentTime(ref __time64_t pCurrentTime, int iDST);
```

Visual Basic

```
Function BSReadCurrentTime(ByRef pCurrentTime As __time64_t, iDST As Integer = -1) As
BODYSTATSDK_API BSError
```

Java

```
BSReadCurrentTime
```

MATLAB

```
function [BODYSTATSDK_API BSError] = BSReadCurrentTime(, )
```

File

BodystatSDK.h (see page 55)

Parameters

Parameters	Description
__time64_t * pCurrentTime	[out] Receives the current time of the device's internal clock.
int iDST = -1	Daylight Saving Time handling (as per Microsoft time functions) Zero (0) to indicate that standard time is in effect. A value greater than 0 to indicate that daylight saving time is in effect. A value less than zero to have the C run-time library code compute whether standard time or daylight saving time is in effect. (default)

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect (see page 11) previously.

1.1.1.1.17 Bodystat::BSReadModelVersion Function

Read device model and firmware version of the connected Bodystat device.

C++

```
BODYSTATSDK_API BSError BSReadModelVersion(BSDeviceModel * pModel, BYTE * pMajorVersion,
BYTE * pMinorVersion, BYTE * pPsoc2Version, BYTE * pEepromVersion, BYTE * pBluetoothInfo);
```

C#

```
BODYSTATSDK_API BSError BSReadModelVersion(ref BSDeviceModel pModel, ref BYTE
pMajorVersion, ref BYTE pMinorVersion, ref BYTE pPsoc2Version, ref BYTE pEepromVersion, ref
BYTE pBluetoothInfo);
```

Visual Basic

```
Function BSReadModelVersion(ByRef pModel As BSDeviceModel, ByRef pMajorVersion As BYTE,
ByRef pMinorVersion As BYTE, ByRef pPsoc2Version As BYTE, ByRef pEepromVersion As BYTE,
ByRef pBluetoothInfo As BYTE) As BODYSTATSDK_API BSError
```

Java

```
BSReadModelVersion
```

MATLAB

```
function [BODYSTATSDK_API BSError] = BSReadModelVersion(, , , , , )
```

File

BodystatSDK.h (see page 55)

Parameters

Parameters	Description
BSDeviceModel * pModel	[out] Receives the device model.
BYTE * pMajorVersion	[out] Receives the major version of the main firmware in the device.
BYTE * pMinorVersion	[out] Receives the minor version of the main firmware in the device.

BYTE * pPsoc2Version	[out] Receives the version of Psoc2 firmware in the device.
BYTE * pEepromVersion	[out] Receives the version of the EEprom firmware in the device.
BYTE * pBluetoothInfo	[out] Receives the version of the Bluetooth module in the device.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect (see page 11) previously.

1.1.1.1.18 Bodystat::BSReadPrinterAddress Function

Read the printer address the connected device is paired with (for direct printing). Not all device models support this function (not all support direct printing).

C++

```
BODYSTATSDK_API BSError BSReadPrinterAddress(LPTSTR lpPrinterAddress, int
iPrinterAddressBufferSize);
```

C#

```
BODYSTATSDK_API BSError BSReadPrinterAddress(LPTSTR lpPrinterAddress, int
iPrinterAddressBufferSize);
```

Visual Basic

```
Function BSReadPrinterAddress(lpPrinterAddress As LPTSTR, iPrinterAddressBufferSize As
Integer) As BODYSTATSDK_API BSError
```

Java

```
BSReadPrinterAddress
```

MATLAB

```
function [BODYSTATSDK_API BSError] = BSReadPrinterAddress(, )
```

File

BodystatSDK.h (see page 55)

Parameters

Parameters	Description
LPTSTR lpPrinterAddress	[out] String buffer which receives the printer address.
int iPrinterAddressBufferSize	Size of the buffer.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect (see page 11) previously.

1.1.1.1.19 Bodystat::BSReadProtocolInfo Function

Read protocol information from the connected device. Contains version numbers relating to communication protocols in use, data structures and auxiliary information.

C++

```
BODYSTATSDK_API BSError BSReadProtocolInfo(BYTE * pProtocolVersion, BYTE * pDataVersion,
BYTE * pAuxInfo);
```

C#

```
BODYSTATSDK_API BSError BSReadProtocolInfo(ref BYTE pProtocolVersion, ref BYTE
pDataVersion, ref BYTE pAuxInfo);
```

Visual Basic

```
Function BSReadProtocolInfo(ByRef pProtocolVersion As BYTE, ByRef pDataVersion As BYTE,
ByRef pAuxInfo As BYTE) As BODYSTATSDK_API BSError
```

Java

```
BSReadProtocolInfo
```

MATLAB

```
function [BODYSTATSDK_API BSError] = BSReadProtocolInfo(, , )
```

File

BodystatSDK.h (see page 55)

Parameters

Parameters	Description
BYTE * pProtocolVersion	[out] Receives the communication protocol version.
BYTE * pDataVersion	[out] Receives the data structure version.
BYTE * pAuxInfo	[out] Receives auxiliary information.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

Generally not required for the majority of SDK users as these details are abstracted by the SDK.

1.1.1.1.20 Bodystat::BSReadSerialNumber Function

Read the unique serial number of the connected Bodystat device.

C++

```
BODYSTATSDK_API BSError BSReadSerialNumber(ULONG * pSerialNumber);
```

C#

```
BODYSTATSDK_API BSError BSReadSerialNumber(ref ULONG pSerialNumber);
```

Visual Basic

```
Function BSReadSerialNumber(ByRef pSerialNumber As ULONG) As BODYSTATSDK_API BSError
```

Java

```
BSReadSerialNumber
```

MATLAB

```
function [BODYSTATSDK_API BSError] = BSReadSerialNumber()
```

File

BodystatSDK.h (see page 55)

Parameters

Parameters	Description
ULONG * pSerialNumber	[out] Receives the serial number of the device.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect (see page 11) previously.

1.1.1.1.21 Bodystat::BSReadStoredTestData Function

Download test data from the connected device. Retrieves the last 100 test measurements (records) stored in the memory of the device. Downloading data does not affect the results stored in the memory - they remain unchanged.

Note: you should store downloaded data locally if it is required permanently. Most devices store only the last 100 test records before overwriting the oldest records automatically.

C++

```
BODYSTATSDK_API BSError BSReadStoredTestData(BSRawData * pRawData);
```

C#

```
BODYSTATSDK_API BSError BSReadStoredTestData(ref BSRawData pRawData);
```

Visual Basic

```
Function BSReadStoredTestData(ByRef pRawData As BSRawData) As BODYSTATSDK_API BSError
```

Java

```
BSReadStoredTestData
```

MATLAB

```
function [BODYSTATSDK_API BSError] = BSReadStoredTestData()
```

File

BodystatSDK.h (see page 55)

Parameters

Parameters	Description
BSRawData * pRawData	[out] Pointer to a structure to receive the downloaded data. Caller responsible for allocation and disposing of the structure.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect (see page 11) previously. Caller must check they are working with a Bodystat (rather than Equistat model, Equistat models will respond without error - but with garbage results).

1.1.1.1.22 Bodystat::BSReadTestBodystat Function

Perform a quick bio-impedance test measurement on the connected bodystat device. Normal test process is skipped (user is not prompted to enter any subject information). Normal test protocol is disregarded (measurement proceeds regardless of lead status). Test measurement is not stored in device memory.

May give wildly erratic readings if leads are not connected properly. Caller should sanity check results to detect for such scenarios. Results are intentionally not sanity checked to provide a true raw reading and provide caller with total discretion.

Useful for diagnostics or performing measurements instigated by software.

C++

```
BODYSTATSDK_API BSError BSReadTestBodystat(UINT * pZ5, UINT * pZ50, UINT * pZ100, UINT *
pZ200, UINT * pR50, float * pX50, float * pPA50);
```

C#

```
BODYSTATSDK_API BSError BSReadTestBodystat(ref UINT pZ5, ref UINT pZ50, ref UINT pZ100, ref
UINT pZ200, ref UINT pR50, ref float pX50, ref float pPA50);
```

Visual Basic

```
Function BSReadTestBodystat(ByRef pZ5 As UINT, ByRef pZ50 As UINT, ByRef pZ100 As UINT,
ByRef pZ200 As UINT, ByRef pR50 As UINT, ByRef pX50 As float, ByRef pPA50 As float) As
BODYSTATSDK_API BSError
```

Java

```
BSReadTestBodystat
```

MATLAB

```
function [BODYSTATSDK_API BSError] = BSReadTestBodystat(, , , , , , )
```

File

BodystatSDK.h (see page 55)

Parameters

Parameters	Description
UINT * pZ5	[out] Receives the impedance measured at 5 kHz (result in ohms).
UINT * pZ50	[out] Receives the impedance measured at 50 kHz (result in ohms).
UINT * pZ100	[out] Receives the impedance measured at 100 kHz (result in ohms).
UINT * pZ200	[out] Receives the impedance measured at 200 kHz (result in ohms).
UINT * pR50	[out] Receives the resistance measured at 50 kHz (result in ohms).
float * pX50	[out] Receives the reactance measured at 50 kHz (result in ohms).
float * pPA50	[out] Receives the phase angle measured at 50 kHz (result in degrees).

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect (see page 11) previously. Caller must check they are working with a Bodystat (rather than Equistat model, Equistat models will respond without error - but with garbage results).

1

1.1.1.1.23 Bodystat::BSSearchBTDevices Function

Search BT radio for any Bodystat devices (paired or otherwise). Existing paired devices will be counted even if they are switched off or out of range. New non-paired units will only be counted if switched on, in range and ready for communication (on main startup screen).

C++

```
BODYSTATSDK_API BOOL BSSearchBTDevices(int & iNewDevices, int & iAuthenticatedDevices, BOOL
bReportErrors = TRUE, HWND hWndParent = NULL, unsigned short iTimeout = 7);
```

C#

```
BODYSTATSDK_API BOOL BSSearchBTDevices(ref int iNewDevices, ref int iAuthenticatedDevices,
BOOL bReportErrors, HWND hWndParent, unsigned short iTimeout);
```

Visual Basic

```
Function BSSearchBTDevices(ByRef iNewDevices As Integer, ByRef iAuthenticatedDevices As
Integer, bReportErrors As BOOL = TRUE, hWndParent As HWND = NULL, iTimeout As unsigned
short = 7) As BODYSTATSDK_API BOOL
```

Java

```
BSSearchBTDevices
```

MATLAB

```
function [BODYSTATSDK_API BOOL] = BSSearchBTDevices(, , , , )
```

File

BodystatSDK.h (see page 55)

Parameters

Parameters	Description
int & iNewDevices	[out] The number of new Bodystat devices found (not paired).
int & iAuthenticatedDevices	[out] The number of authenticated Bodystat devices (paired).
BOOL bReportErrors = TRUE	TRUE to report errors to the user (shown as message boxes, locale constraints apply). FALSE to hide errors from the user (completely silent).
HWND hWndParent = NULL	Handle of the parent window.
unsigned short iTimeout = 7	Timeout multiplier supplied to the Bluetooth radio search API. Controls the amount of time to spend searching for the device. This function will block for this duration so is better called from a thread. Each multiplier increment equates to about 1.28 seconds. Thus a timeout value of 10 = 12.8s. Maximum time permitted by the API is a multiplier of 45 (just under 1 minute), values higher than this will be capped at 45. Default value is 7 (about 10 seconds).

Returns

TRUE if it succeeds, FALSE if it fails.

Remarks

Note this process can be time consuming to complete and executes synchronously. Either call from a threaded function or illustrate operation in progress to the user e.g. by means of an indefinite progress bar or busy mouse cursor, until the task completes or timesout.

1.1.1.1.24 Bodystat::BSUnAuthenticateBTDevices Function

Unauthenticate (unpair) ALL Bodystat devices from this PC.

C++

```
BODYSTATSDK_API BOOL BSUnAuthenticateBTDevices(BOOL bReportErrors = TRUE, HWND hWndParent =
NULL, unsigned short iTimeout = 7);
```

C#

```
BODYSTATSDK_API BOOL BSUnAuthenticateBTDevices(BOOL bReportErrors, HWND hWndParent,
unsigned short iTimeout);
```

Visual Basic

```
Function BSUnAuthenticateBTDevices(bReportErrors As BOOL = TRUE, hWndParent As HWND = NULL,
iTimeout As unsigned short = 7) As BODYSTATSDK_API BOOL
```

Java

```
BSUnAuthenticateBTDevices
```

MATLAB

```
function [BODYSTATSDK_API BOOL] = BSUnAuthenticateBTDevices(, , )
```

File

BodystatSDK.h (see page 55)

Parameters

Parameters	Description
BOOL bReportErrors = TRUE	TRUE to report errors to the user (shown as message boxes, locale constraints apply). FALSE to hide errors from the user.
HWND hWndParent = NULL	Handle of the parent window.
unsigned short iTimeout = 7	Timeout multiplier supplied to the Bluetooth radio search API. Controls the amount of time to spend searching for the device. This function will block for this duration so is better called from a thread. Each multiplier increment equates to about 1.28 seconds. Thus a timeout value of 10 = 12.8s. Maximum time permitted by the API is a multiplier of 45 (just under 1 minute), values higher than this will be capped at 45. Default value is 7 (about 10 seconds).

Returns

TRUE if it succeeds, FALSE if it fails.

Remarks

Note this process can be time consuming to complete and executes synchronously. Either call from a threaded function or illustrate operation in progress to the user e.g. by means of an indefinite progress bar or busy mouse cursor, until the task completes or timesout.

1.1.1.1.25 Bodystat::BSWriteCurrentTime Function

Reset the current date/time of the internal clock on the connected device. Date/time is automatically set to match the current date/time of the PC

C++

```
BODYSTATSDK_API BSError BSWriteCurrentTime();
```

C#

```
BODYSTATSDK_API BSError BSWriteCurrentTime();
```

Visual Basic

```
Function BSWriteCurrentTime() As BODYSTATSDK_API BSError
```

Java

```
BSWriteCurrentTime
```

MATLAB

```
function [BODYSTATSDK_API BSError] = BSWriteCurrentTime()
```

File

BodystatSDK.h (see page 55)

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect (see page 11) previously. Locale Daylight Saving Time (DST) rules are observed. Use sister function BSWriteCurrentTime(__time64_t CurrentTime) to set a specific date/time which differs from the PC clock or override DST behaviour.

1.1.1.1.26 Bodystat::BSWriteCurrentTimeAs Function

Reset the current date/time of the internal clock on the connected device. Date/time is set to a specific value of your choosing.

C++

```
BODYSTATSDK_API BSError BSWriteCurrentTimeAs(__time64_t CurrentTime);
```

C#

```
BODYSTATSDK_API BSError BSWriteCurrentTimeAs(__time64_t CurrentTime);
```

Visual Basic

```
Function BSWriteCurrentTimeAs(CurrentTime As __time64_t) As BODYSTATSDK_API BSError
```

Java

```
BSWriteCurrentTimeAs
```

MATLAB

```
function [BODYSTATSDK_API BSError] = BSWriteCurrentTimeAs()
```

File

BodystatSDK.h (see page 55)

Parameters

Parameters	Description
__time64_t CurrentTime	Date/time to apply to the internal clock of the device (should be current!).

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect (see page 11) previously. Use sister function (omit date/time parameter entirely) BSWriteCurrentTime (see page 24)() to automatically set to the current date/time of the PC.

1.1.1.1.27 Bodystat::BSWritePrinterAddress Function

Set the address of the printer the connected device should use for direct printing. Programmatic implementation of the inbuilt printer pairing function in the device.

C++

```
BODYSTATSDK_API BSError BSWritePrinterAddress(LPCTSTR lpPrinterAddress, int iBufferSize);
```

C#

```
BODYSTATSDK_API BSError BSWritePrinterAddress(LPCTSTR lpPrinterAddress, int iBufferSize);
```

Visual Basic

```
Function BSWritePrinterAddress(lpPrinterAddress As LPCTSTR, iBufferSize As Integer) As
BODYSTATSDK_API BSError
```

Java

```
BSWritePrinterAddress
```

MATLAB

```
function [BODYSTATSDK_API BSError] = BSWritePrinterAddress(, )
```

File

BodystatSDK.h (see page 55)

Parameters

Parameters	Description
LPCTSTR lpPrinterAddress	String buffer containing the Bluetooth address (BDA) of the printer.
int iBufferSize	Size of the buffer.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

Note not all device models support this function (not all support direct printing).

1.1.1.1.28 Bodystat::ESCalculateNormals Function

Calculate normal values and normal ranges for a given animal and their bio-impedance measurement together with the matching results structure for that measurement. You must supply a populated results structure which pertains to the measurement being supplied (obtained from ESCalculateResults (see page 26)).

C++

```
BODYSTATSDK_API BSError ESCalculateNormals(const ESMeasurement * pM, const ESResults * pR,
ESNormals * pN);
```

C#

```
BODYSTATSDK_API BSError ESCalculateNormals(ESMeasurement * pM, ESResults * pR, ref
ESNormals pN);
```

Visual Basic

```
Function ESCalculateNormals(pM As ESMeasurement *, pR As ESResults *, ByRef pN As
ESNormals) As BODYSTATSDK_API BSError
```

Java

```
ESCalculateNormals
```

MATLAB

```
function [BODYSTATSDK_API BSError] = ESCalculateNormals(, , )
```

File

BodystatSDK.h (see page 55)

Parameters

Parameters	Description
const ESMeasurement * pM	The measurement details downloaded from the device (includes model, user inputted parameters and bio-impedance measurement readings).
const ESResults * pR	The results calculated for this measurement previously.
ESNormals * pN	[out] Receives the normal values and normal ranges for this measurement.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

Normals vary by input attributes, measured bio-impedance values and calculated results.

1.1.1.1.29 Bodystat::ESCalculateResults Function

Calculate complete result set for a given bio-impedance measurement recorded by the Equistat device.

C++

```
BODYSTATSDK_API BSError ESCalculateResults(const ESMeasurement * pM, ESResults * pR);
```

C#

```
BODYSTATSDK_API BSError ESCalculateResults(ESMeasurement * pM, ref ESResults pR);
```

Visual Basic

```
Function ESCalculateResults(pM As ESMeasurement *, ByRef pR As ESResults) As
BODYSTATSDK_API BSError
```

Java

```
ESCalculateResults
```

MATLAB

```
function [BODYSTATSDK_API BSError] = ESCalculateResults(, )
```

File

BodystatSDK.h (see page 55)

Parameters

Parameters	Description
const ESMeasurement * pM	The measurement details downloaded from the device (includes model, user inputted parameters and bio-impedance measurement readings).
ESResults * pR	[out] Receives the complete calculated results for this measurement.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

Results vary by input attributes and measured bio-impedance values.

1.1.1.1.30 Bodystat::ESReadStoredTestData Function

Download test data from the connected device. Retrieves the last 100 test measurements (records) stored in the memory of the device. Downloading data does not affect the results stored in the memory - they remain unchanged.

Note: you should store downloaded data locally if it is required permanently. Most devices store only the last 100 test records before overwriting the oldest records automatically.

C++

```
BODYSTATSDK_API BSError ESReadStoredTestData(ESRawData * pRawData);
```

C#

```
BODYSTATSDK_API BSError ESReadStoredTestData(ref ESRawData pRawData);
```

Visual Basic

```
Function ESReadStoredTestData(ByRef pRawData As ESRawData) As BODYSTATSDK_API BSError
```

Java

```
ESReadStoredTestData
```

MATLAB

```
function [BODYSTATSDK_API BSError] = ESReadStoredTestData()
```

File

BodystatSDK.h (see page 55)

Parameters

Parameters	Description
ESRawData * pRawData	[out] Pointer to a structure to receive the downloaded data. Caller responsible for allocation and disposing of the structure.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect (see page 11) previously. Caller must check they are working with an Equistat (rather than Bodystat model, Bodystat models will respond without error - but with garbage results).

1.1.1.1.31 Bodystat::ESReadTestEquistat Function

Perform a quick bio-impedance test measurement on the connected equistat device. Normal test process is skipped (user is not prompted to enter any horse information). Normal test protocol is disregarded (measurement proceeds regardless of lead status). Test measurement is not stored in device memory.

May give wildly erratic readings if leads are not connected properly. Caller should sanity check results to detect for such scenarios. Results are intentionally not sanity checked to provide a true raw reading and provide caller with total discretion.

Useful for diagnostics or performing measurements instigated by software.

C++

```
BODYSTATSDK_API BSError ESReadTestEquistat(UINT * pZ5, UINT * pZ16, UINT * pZ24, UINT *
pZ50, UINT * pZ140, UINT * pZ200, UINT * pZ280, float * pPA50);
```

C#

```
BODYSTATSDK_API BSError ESReadTestEquistat(ref UINT pZ5, ref UINT pZ16, ref UINT pZ24, ref
UINT pZ50, ref UINT pZ140, ref UINT pZ200, ref UINT pZ280, ref float pPA50);
```

Visual Basic

```
Function ESReadTestEquistat(ByRef pZ5 As UINT, ByRef pZ16 As UINT, ByRef pZ24 As UINT,
ByRef pZ50 As UINT, ByRef pZ140 As UINT, ByRef pZ200 As UINT, ByRef pZ280 As UINT, ByRef
pPA50 As float) As BODYSTATSDK_API BSError
```

Java

```
ESReadTestEquistat
```

MATLAB

```
function [BODYSTATSDK_API BSError] = ESReadTestEquistat(, , , , , , , )
```

File

BodystatSDK.h (see page 55)

Parameters

Parameters	Description
UINT * pZ5	[out] Receives the impedance measured at 5 kHz (result in ohms).
UINT * pZ16	[out] Receives the impedance measured at 16 kHz (result in ohms).
UINT * pZ24	[out] Receives the impedance measured at 24 kHz (result in ohms).
UINT * pZ50	[out] Receives the impedance measured at 50 kHz (result in ohms).
UINT * pZ140	[out] Receives the impedance measured at 140 kHz (result in ohms).
UINT * pZ200	[out] Receives the impedance measured at 200 kHz (result in ohms).
UINT * pZ280	[out] Receives the impedance measured at 280 kHz (result in ohms).
float * pPA50	[out] Receives the phase angle measured at 50 kHz (result in degress) (ES Pro Only).

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect (see page 11) previously. Caller must check they are working with an Equistat (rather than Bodystat model, Bodystat models will respond without error - but with garbage results).

1.1.1.2 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description

	BSAnalysisMode (see page 30)	BSAnalysisMode: Analysis results are available in several formats:
		BSAnalysisModeBC= Body composition analysis
		BSAnalysisModeHN= Hydration analysis
		BSAnalysisModeBoth= Combined body composition and hydration analysis (default)

	BSDeviceFamily (see page 30)	BSDeviceFamily: Enumerated values that group each device / model variant into a given product family. Devices within a given product family share broad capabilities and features. For details regarding capabilities and features refer to the technical documentation for that product family (http://www.bodystat.com)
	BSDeviceModel (see page 32)	BSDeviceModel: Enumerated values that represent the various different models of the Bodystat devices over the years. Suffix: The suffix after the model indicates specific hardware variations regarding connectivity: OPTO indicates model communicates via opto-isolated serial. DIU indicates model communicates via a data interface unit (induction interface) BT indicates model communicates via Bluetooth serial port.
	BSError (see page 33)	BSError: Bodystat Error codes. A list of Bodystat specific error codes.
	BSGender (see page 35)	BSGender: Enumerated values that represent a person's gender.

Structures

	Name	Description
	BSMeasurement (see page 36)	BSMeasurement: Measurement structure. Each test measurement recorded in the device is stored one of these structures. Holds the user inputted information about the subject, together with bio-impedance readings measured by the device (raw electrical readings only, not full results).
	BSNormals (see page 38)	BSNormals: Bodystat Normals structure. Holds the normal values or normal ranges for any given test measurement.
	BSRawData (see page 40)	BSRawData: Bodystat Raw Data structure. Used when downloading test measurements from the device. Contains a fixed array of measurement records together with information about the number of records occupied within the array and test record starting number.
	BSResults (see page 41)	BSResults: Bodystat Results structure. Holds the calculated results for any given test measurement.
	ESMeasurement (see page 44)	ESMeasurement: Equistat Measurement structure. Each test measurement recorded in the device is stored one of these structures. Holds the user inputted information about the horse, together with bio-impedance readings measured by the device (raw electrical readings only, not full results).
	ESNormals (see page 45)	ESNormals: Equistat Normals structure. Holds the normal values or normal ranges for any given test measurement.

◆	ESRawData (see page 47)	ESRawData: Equistat Raw Data structure. Used when downloading test measurements from the device. Contains a fixed array of measurement records together with information about the number of records occupied within the array and test record starting number.
◆	ESResults (see page 48)	ESResults: Equistat Results structure. Holds the calculated results for any given test measurement.
◆	ImpData (see page 49)	Represents one frequency measurement

1.1.1.2.1 Bodystat::BSAnalysisMode Enumeration

BSAnalysisMode: Analysis results are available in several formats:

BSAnalysisModeBC= Body composition analysis

BSAnalysisModeHN= Hydration analysis

BSAnalysisModeBoth= Combined body composition and hydration analysis (default)

C++

```
enum BSAnalysisMode {
  BSAnalysisModeNone = 0,
  BSAnalysisModeBC = 1,
  BSAnalysisModeHN = 2,
  BSAnalysisModeBoth = 3
};
```

C#

```
public enum BSAnalysisMode {
  BSAnalysisModeNone = 0,
  BSAnalysisModeBC = 1,
  BSAnalysisModeHN = 2,
  BSAnalysisModeBoth = 3
}
```

Visual Basic

```
Public Enum BSAnalysisMode
  BSAnalysisModeNone = 0
  BSAnalysisModeBC = 1
  BSAnalysisModeHN = 2
  BSAnalysisModeBoth = 3
End Enum
```

Java

```
public class BSAnalysisMode;
```

MATLAB

```
BSAnalysisMode
```

File

BodystatSDK.h (see page 55)

Members

Members	Description
BSAnalysisModeNone = 0	No analysis mode specified (raw data only)
BSAnalysisModeBC = 1	Body composition enabled
BSAnalysisModeHN = 2	Hydration enabled
BSAnalysisModeBoth = 3	Both body composition and hydration (default)

1.1.1.2.2 Bodystat::BSDeviceFamily Enumeration

BSDeviceFamily: Enumerated values that group each device / model variant into a given product family. Devices within a given product family share broad capabilities and features. For details regarding capabilities and features refer to the technical documentation for that product family (http://www.bodystat.com)

C++

```
enum BSDeviceFamily {
  BSUnknown_Family = 0,
  BS1500_Family = -1,
  BSMDD_Family = -2,
  BSDualScan_Family = -3,
  BSQuadScan_Family = -4,
  BSMultiScan_Family = -5,
  BSEquistatLite_Family = -6,
  BSEquistatPro_Family = -7
};
```

C#

```
public enum BSDeviceFamily {
  BSUnknown_Family = 0,
  BS1500_Family = -1,
  BSMDD_Family = -2,
  BSDualScan_Family = -3,
  BSQuadScan_Family = -4,
  BSMultiScan_Family = -5,
  BSEquistatLite_Family = -6,
  BSEquistatPro_Family = -7
}
```

Visual Basic

```
Public Enum BSDeviceFamily
  BSUnknown_Family = 0
  BS1500_Family = -1
  BSMDD_Family = -2
  BSDualScan_Family = -3
  BSQuadScan_Family = -4
  BSMultiScan_Family = -5
  BSEquistatLite_Family = -6
  BSEquistatPro_Family = -7
End Enum
```

Java

```
public class BSDeviceFamily;
```

MATLAB

```
BSDeviceFamily
```

File

BodystatSDK.h (see page 55)

Members

Members	Description
BSUnknown_Family = 0	Unknown product family
BS1500_Family = -1	Bodystat 1500 family
BSMDD_Family = -2	Bodystat MDD family
BSDualScan_Family = -3	Bodystat DualScan family (legacy)
BSQuadScan_Family = -4	Bodystat QuadScan family
BSMultiScan_Family = -5	Bodystat Multiscan family
BSEquistatLite_Family = -6	Equistat Lite family (not supported by this SDK)
BSEquistatPro_Family = -7	Equistat Pro family (not supported by this SDK)

Remarks

Note that not all families are supported by this SDK. Some are listed for historic reference only.

1.1.1.2.3 Bodystat::BSDeviceModel Enumeration

BSDeviceModel: Enumerated values that represent the various different models of the Bodystat devices over the years.

Suffix: The suffix after the model indicates specific hardware variations regarding connectivity:

OPTO indicates model communicates via opto-isolated serial.

DIU indicates model communicates via a data interface unit (induction interface)

BT indicates model communicates via Bluetooth serial port.

C++

```
enum BSDeviceModel {
  BSUnknown = 0,
  BS1500_OPTO = 1,
  BSMDD_OPTO = 2,
  BS1500_DIU = 3,
  BSMDD_DIU = 4,
  BSDualScan_OPTO = 5,
  BSDualScan_DIU = 6,
  BSQuadScan_DIU = 7,
  BSMultiScan_DIU = 8,
  BS1500_BT = 9,
  BSMDD_BT = 10,
  BSQuadScan_BT = 11,
  BSEquistatPro_BT = 12,
  BSEquistatLite_BT = 13,
  BSQuadScanTouch = 14,
  BSMultiScanTouch = 15
};
```

C#

```
public enum BSDeviceModel {
  BSUnknown = 0,
  BS1500_OPTO = 1,
  BSMDD_OPTO = 2,
  BS1500_DIU = 3,
  BSMDD_DIU = 4,
  BSDualScan_OPTO = 5,
  BSDualScan_DIU = 6,
  BSQuadScan_DIU = 7,
  BSMultiScan_DIU = 8,
  BS1500_BT = 9,
  BSMDD_BT = 10,
  BSQuadScan_BT = 11,
  BSEquistatPro_BT = 12,
  BSEquistatLite_BT = 13,
  BSQuadScanTouch = 14,
  BSMultiScanTouch = 15
}
```

Visual Basic

```
Public Enum BSDeviceModel
  BSUnknown = 0
  BS1500_OPTO = 1
  BSMDD_OPTO = 2
  BS1500_DIU = 3
  BSMDD_DIU = 4
  BSDualScan_OPTO = 5
  BSDualScan_DIU = 6
  BSQuadScan_DIU = 7
  BSMultiScan_DIU = 8
  BS1500_BT = 9
  BSMDD_BT = 10
  BSQuadScan_BT = 11
  BSEquistatPro_BT = 12
  BSEquistatLite_BT = 13
  BSQuadScanTouch = 14
  BSMultiScanTouch = 15
End Enum
```

Java

```
public class BSDeviceModel;
```

MATLAB

```
BSDeviceModel
```

File

BodystatSDK.h (see page 55)

Members

Members	Description
BSUnknown = 0	Unknown model
BS1500_OPTO = 1	Bodystat 1500 with opto-isolated interface (legacy)
BSMDD_OPTO = 2	Bodystat MDD with opto-isolated interface (legacy)
BS1500_DIU = 3	Bodystat 1500 with data interface unit (legacy)
BSMDD_DIU = 4	Bodystat MDD with data interface unit (legacy)
BSDualScan_OPTO = 5	Bodystat DualScan with opto-isolated interface (legacy)
BSDualScan_DIU = 6	Bodystat DualScan with data interface unit (legacy)
BSQuadScan_DIU = 7	Bodystat QuadScan with data interface unit (legacy)
BSMultiScan_DIU = 8	Bodystat MultiScan with data interface unit (legacy)
BS1500_BT = 9	Bodystat 1500 with Bluetooth interface
BSMDD_BT = 10	Bodystat 1500MDD with Bluetooth interface
BSQuadScan_BT = 11	Bodystat QuadScan 4000 with Bluetooth interface
BSEquistatPro_BT = 12	Equistat Pro with Bluetooth interface (not supported by this SDK)
BSEquistatLite_BT = 13	Equistat Lite with Bluetooth interface (not supported by this SDK)
BSQuadScanTouch = 14	Bodystat QuadScan 4000 Touch with Wifi/Bluetooth interface
BSMultiScanTouch = 15	Bodystat Multiscan with Wifi/Bluetooth interface

Remarks

Note that not all models are supported by this SDK. Some are listed for historic reference only.

1.1.1.2.4 Bodystat::BSError Enumeration

BSError: Bodystat Error codes. A list of Bodystat specific error codes.

C++

```
enum BSError {
  NoError = 0,
  InvalidDeviceModel = -1,
  InvalidDeviceFamily = -2,
  InvalidDeviceMode = -3,
  InvalidGender = -4,
  InvalidAge = -5,
  InvalidHeight = -6,
  InvalidWeight = -7,
  InvalidActivity = -8,
  InvalidWaist = -9,
  InvalidHip = -10,
  InvalidImpedanceZ5 = -11,
  InvalidImpedanceZ50 = -12,
  InvalidImpedanceZ100 = -13,
  InvalidImpedanceZ200 = -14,
  InvalidResistanceR50 = -15,
  InvalidReactanceX50 = -16,
  InvalidPhaseAnglePA50 = -17,
  InvalidParamSupplied = -18,
  ComErrorOpeningPort = -50,
  ComErrorConfiguringPort = -51,
  ComErrorConnecting = -52,
  ComErrorUnsupportedCommand = -53,
  ComErrorProcessingCommand = -54,
  ComErrorReading = -55,
  ComErrorWriting = -56,
  ComErrorUserAborted = -57
};
```

C#

```
  public enum BSError {
    NoError = 0,
    InvalidDeviceModel = -1,
    InvalidDeviceFamily = -2,
    InvalidDeviceMode = -3,
    InvalidGender = -4,
    InvalidAge = -5,
    InvalidHeight = -6,
    InvalidWeight = -7,
    InvalidActivity = -8,
    InvalidWaist = -9,
    InvalidHip = -10,
    InvalidImpedanceZ5 = -11,
    InvalidImpedanceZ50 = -12,
    InvalidImpedanceZ100 = -13,
    InvalidImpedanceZ200 = -14,
    InvalidResistanceR50 = -15,
    InvalidReactanceX50 = -16,
    InvalidPhaseAnglePA50 = -17,
    InvalidParamSupplied = -18,
    ComErrorOpeningPort = -50,
    ComErrorConfiguringPort = -51,
    ComErrorConnecting = -52,
    ComErrorUnsupportedCommand = -53,
    ComErrorProcessingCommand = -54,
    ComErrorReading = -55,
    ComErrorWriting = -56,
    ComErrorUserAborted = -57
  }
```

Visual Basic

```
  Public Enum BSError
    NoError = 0
    InvalidDeviceModel = -1
    InvalidDeviceFamily = -2
    InvalidDeviceMode = -3
    InvalidGender = -4
    InvalidAge = -5
    InvalidHeight = -6
    InvalidWeight = -7
    InvalidActivity = -8
    InvalidWaist = -9
    InvalidHip = -10
    InvalidImpedanceZ5 = -11
    InvalidImpedanceZ50 = -12
    InvalidImpedanceZ100 = -13
    InvalidImpedanceZ200 = -14
    InvalidResistanceR50 = -15
    InvalidReactanceX50 = -16
    InvalidPhaseAnglePA50 = -17
    InvalidParamSupplied = -18
    ComErrorOpeningPort = -50
    ComErrorConfiguringPort = -51
    ComErrorConnecting = -52
    ComErrorUnsupportedCommand = -53
    ComErrorProcessingCommand = -54
    ComErrorReading = -55
    ComErrorWriting = -56
    ComErrorUserAborted = -57
  End Enum
```

Java

```
public class BSError;
```

MATLAB

```
BSError
```

File

BodystatSDK.h (see page 55)

Members

Members	Description
NoError = 0	No error occurred
InvalidDeviceModel = -1	Invalid model specified
InvalidDeviceFamily = -2	Invalid family specified
InvalidDeviceMode = -3	Invalid analysis mode
InvalidGender = -4	Invalid BSGender (see page 35) specified
InvalidAge = -5	Invalid age specified
InvalidHeight = -6	Invalid height measurement specified
InvalidWeight = -7	Invalid weight measurement specified
InvalidActivity = -8	Invalid activity level specified
InvalidWaist = -9	Invalid waist measurement specified
InvalidHip = -10	Invalid hip measurement specified
InvalidImpedanceZ5 = -11	Invalid impedance measurement specified
InvalidImpedanceZ50 = -12	Invalid impedance measurement specified
InvalidImpedanceZ100 = -13	Invalid impedance measurement specified
InvalidImpedanceZ200 = -14	Invalid impedance measurement specified
InvalidResistanceR50 = -15	Invalid resistance measurement specified
InvalidReactanceX50 = -16	Invalid reactance measurement specified
InvalidPhaseAnglePA50 = -17	Invalid phase angle measurement specified
InvalidParamSupplied = -18	Invalid parameter supplied
ComErrorOpeningPort = -50	Error opening the com port
ComErrorConfiguringPort = -51	Error configuring the com port parameters (required baud rate, buffers, stop bits, etc)
ComErrorConnecting = -52	Error connecting to the device
ComErrorUnsupportedCommand = -53	The requested command is not supported by the model being asked to perform it
ComErrorProcessingCommand = -54	Error occurred whilst processing the requested command
ComErrorReading = -55	Error occurred whilst reading data from the com port
ComErrorWriting = -56	Error occurred whilst writing data to the com port
ComErrorUserAborted = -57	The user aborted the process

1.1.1.2.5 Bodystat::BSGender Enumeration

BSGender: Enumerated values that represent a person's gender.

C++

```
enum BSGender {
  BSFemale = 0,
  BSMale = 1
};
```

C#

```
public enum BSGender {
  BSFemale = 0,
  BSMale = 1
}
```

Visual Basic

```
Public Enum BSGender
  BSFemale = 0
  BSMale = 1
End Enum
```

Java

```
public class BSGender;
```

MATLAB

```
BSGender
```

File

BodystatSDK.h (see page 55)

Members

Members	Description
BSFemale = 0	Female
BSMale = 1	Male

1.1.1.2.6 Bodystat::BSMeasurement Structure

BSMeasurement: Measurement structure. Each test measurement recorded in the device is stored one of these structures. Holds the user inputted information about the subject, together with bio-impedance readings measured by the device (raw electrical readings only, not full results).

C++

```
struct BSMeasurement {
  BSDeviceModel iDeviceModel;
  ULONG ulDeviceSerialNumber;
  __time64_t tTestDate;
  BSGender iGender;
  int iAge;
  int iHeight;
  float fWeight;
  int iActivity;
  int iWaist;
  int iHip;
  int iZ_5kHz;
  int iZ_50kHz;
  int iZ_100kHz;
  int iZ_200kHz;
  int iR_50kHz;
  float fX_50kHz;
  float fPA_50kHz;
  int iFrequencies;
  ImpData* pMultifreqData;
};
```

C#

```
public struct BSMeasurement {
  public BSDeviceModel iDeviceModel;
  public ULONG ulDeviceSerialNumber;
  public __time64_t tTestDate;
  public BSGender iGender;
  public int iAge;
  public int iHeight;
  public float fWeight;
```

```
    public int iActivity;
    public int iWaist;
    public int iHip;
    public int iZ_5kHz;
    public int iZ_50kHz;
    public int iZ_100kHz;
    public int iZ_200kHz;
    public int iR_50kHz;
    public float fX_50kHz;
    public float fPA_50kHz;
    public int iFrequencies;
    public ImpData* pMultifreqData;
  }
```

Visual Basic

```
Public Structure BSMeasurement
  Public iDeviceModel As BSDeviceModel
  Public ulDeviceSerialNumber As ULONG
  Public tTestDate As __time64_t
  Public iGender As BSGender
  Public iAge As Integer
  Public iHeight As Integer
  Public fWeight As float
  Public iActivity As Integer
  Public iWaist As Integer
  Public iHip As Integer
  Public iZ_5kHz As Integer
  Public iZ_50kHz As Integer
  Public iZ_100kHz As Integer
  Public iZ_200kHz As Integer
  Public iR_50kHz As Integer
  Public fX_50kHz As float
  Public fPA_50kHz As float
  Public iFrequencies As Integer
  Public pMultifreqData As ImpData*
End Structure
```

Java

```
public class BSMeasurement;
```

MATLAB

```
BSMeasurement
```

File

BodystatSDK.h (see page 55)

Members

Members	Description
BSDeviceModel iDeviceModel;	Device model which performed the measurement
ULONG ulDeviceSerialNumber;	Unique serial number of the device which performed the measurement
__time64_t tTestDate;	Date & time the test measurement was performed (Windows 64-bit time structure) (MDD/Quad only)
BSGender iGender;	Inputted gender of the subject
int iAge;	Inputted age of the subject (years)
int iHeight;	Inputted height of the subject (centimeters)
float fWeight;	Inputted weight of the subject (kilograms)
int iActivity;	Inputted physical activity level of the subject (range 1 to 5: 1=very low 2=low/medium 3=medium 4=medium/high 5=very high)
int iWaist;	Inputted waist size of the subject (centimeters)
int iHip;	Inputted hip size of the subject (centimeters)
int iZ_5kHz;	Impedance measured at 5 kHz (result in ohms) (MDD/Quad only)

int iZ_50kHz;	Impedance measured at 50 kHz (result in ohms)
int iZ_100kHz;	Impedance measured at 100 kHz (result in ohms) (Quad only)
int iZ_200kHz;	Impedance measured at 200 kHz (result in ohms) (Quad only)
int iR_50kHz;	Resistance measured at 50 kHz (result in ohms) (MDD/Quad only)
float fX_50kHz;	Reactance measured at 50 kHz (result in ohms) (MDD/Quad only)
float fPA_50kHz;	Phase angle measured at 50 kHz (result in degrees) (MDD/Quad only)
int iFrequencies;	number of frequencies used 11/50
ImpData* pMultifreqData;	pointer to ImpData (see page 49) array

1.1.1.2.7 Bodystat::BSNormals Structure

BSNormals: Bodystat Normals structure. Holds the normal values or normal ranges for any given test measurement.

C++

```
struct BSNormals {
  int iFatPerc_L;
  int iFatPerc_H;
  int iFatKg_L;
  int iFatKg_H;
  int iLeanPerc_L;
  int iLeanPerc_H;
  int iLeanKg_L;
  int iLeanKg_H;
  int iTotalWeight_L;
  int iTotalWeight_H;
  int iTotalWeightMethod;
  int iTBWPerc_L;
  int iTBWPerc_H;
  int iTBW_L;
  int iTBW_H;
  int iECWPerc_Norm;
  int iICWPerc_Norm;
  int iBFMI_L;
  int iBFMI_H;
  int iFFMI_L;
  int iFFMI_H;
  float fNutrition_Norm;
  float fIllness_L;
  float fIllness_H;
  int iBMI_L;
  int iBMI_H;
  float fWaistHip_Norm;
  float fWellness_L;
  float fWellness_H;
};
```

C#

```
public struct BSNormals {
  public int iFatPerc_L;
  public int iFatPerc_H;
  public int iFatKg_L;
  public int iFatKg_H;
  public int iLeanPerc_L;
  public int iLeanPerc_H;
  public int iLeanKg_L;
  public int iLeanKg_H;
  public int iTotalWeight_L;
  public int iTotalWeight_H;
  public int iTotalWeightMethod;
  public int iTBWPerc_L;
```

```
    public int iTBWPerc_H;
    public int iTBW_L;
    public int iTBW_H;
    public int iECWPerc_Norm;
    public int iICWPerc_Norm;
    public int iBFMI_L;
    public int iBFMI_H;
    public int iFFMI_L;
    public int iFFMI_H;
    public float fNutrition_Norm;
    public float fIllness_L;
    public float fIllness_H;
    public int iBMI_L;
    public int iBMI_H;
    public float fWaistHip_Norm;
    public float fWellness_L;
    public float fWellness_H;
  }
```

Visual Basic

```
  Public Structure BSNormals
    Public iFatPerc_L As Integer
    Public iFatPerc_H As Integer
    Public iFatKg_L As Integer
    Public iFatKg_H As Integer
    Public iLeanPerc_L As Integer
    Public iLeanPerc_H As Integer
    Public iLeanKg_L As Integer
    Public iLeanKg_H As Integer
    Public iTotalWeight_L As Integer
    Public iTotalWeight_H As Integer
    Public iTotalWeightMethod As Integer
    Public iTBWPerc_L As Integer
    Public iTBWPerc_H As Integer
    Public iTBW_L As Integer
    Public iTBW_H As Integer
    Public iECWPerc_Norm As Integer
    Public iICWPerc_Norm As Integer
    Public iBFMI_L As Integer
    Public iBFMI_H As Integer
    Public iFFMI_L As Integer
    Public iFFMI_H As Integer
    Public fNutrition_Norm As float
    Public fIllness_L As float
    Public fIllness_H As float
    Public iBMI_L As Integer
    Public iBMI_H As Integer
    Public fWaistHip_Norm As float
    Public fWellness_L As float
    Public fWellness_H As float
  End Structure
```

Java

```
  public class BSNormals;
```

MATLAB

```
  BSNormals
```

File

BodystatSDK.h (see page 55)

Members

Members	Description
int iFatPerc_L;	Normal range for fat percentage - (low end)
int iFatPerc_H;	Normal range for fat percentage - (high end)
int iFatKg_L;	Normal range for fat (in kg) - (low end)
int iFatKg_H;	Normal range for fat (in kg) - (high end)

int iLeanPerc_L;	Normal range for lean percentage - (low end)
int iLeanPerc_H;	Normal range for lean percentage - (high end)
int iLeanKg_L;	Normal range for lean (in kg) - (low end)
int iLeanKg_H;	Normal range for lean (in kg) - (high end)
int iTotalWeight_L;	Normal range for Total Weight (in kg) - (low end)
int iTotalWeight_H;	Normal range for Total Weight (in kg) - (high end)
int iTotalWeightMethod;	Method used when calculating Total Weight normal range (0 = Composition Method, 1 = BMI Method)
int iTBWPerc_L;	Normal range for Total Body Water percentage - (low end)
int iTBWPerc_H;	Normal range for Total Body Water percentage - (high end)
int iTBW_L;	Normal range for Total Body Water (in litres) - (low end)
int iTBW_H;	Normal range for Total Body Water (in litres) - (high end)
int iECWPerc_Norm;	Normal value for ECW Percentage (Quad only)
int iICWPerc_Norm;	Normal value for ICW Percentage (Quad only)
int iBFMI_L;	Normal range for Body Fat Mass Index (MDD/Quad only) - (low end)
int iBFMI_H;	Normal range for Body Fat Mass Index (MDD/Quad only) - (high end)
int iFFMI_L;	Normal range for Fat Free Mass Index (MDD/Quad only) - (low end)
int iFFMI_H;	Normal range for Fat Free Mass Index (MDD/Quad only) - (high end)
float fNutrition_Norm;	Normal value for Nutrition Index (Quad only)
float fIllness_L;	Normal range for Prediction Marker (TM) - formerly known as Illness Marker (TM) - (Quad only) - (low end)
float fIllness_H;	Normal range for Prediction Marker (TM) - formerly known as Illness Marker (TM) - (Quad only) - (high end)
int iBMI_L;	Normal range for Body Mass Index - (low end)
int iBMI_H;	Normal range for Body Mass Index - (high end)
float fWaistHip_Norm;	Normal value for Waist/Hip ratio
float fWellness_L;	Normal range for Wellness Marker (TM) (MDD only) - (low end)
float fWellness_H;	Normal range for Wellness Marker (TM) (MDD only) - (high end)

1.1.1.2.8 Bodystat::BSRawData Structure

BSRawData: Bodystat Raw Data structure. Used when downloading test measurements from the device. Contains a fixed array of measurement records together with information about the number of records occupied within the array and test record starting number.

C++

```
struct BSRawData {
  int iRecordArraySize;
  BSMeasurement record[BS_RAWDATA_ARRAYSIZE];
  int iTotalNumRecs;
  int ulFirstTestNum;
};
```

C#

```
public struct BSRawData {
  public int iRecordArraySize;
  public BSMeasurement record;
  public int iTotalNumRecs;
  public int ulFirstTestNum;
}
```

Visual Basic

```
Public Structure BSRawData
  Public iRecordArraySize As Integer
  Public record As BSMeasurement
  Public iTotalNumRecs As Integer
  Public ulFirstTestNum As Integer
End Structure
```

Java

```
public class BSRawData;
```

MATLAB

```
BSRawData
```

File

BodystatSDK.h (see page 55)

Members

Members	Description
int iRecordArraySize;	Maximum number of records this structure has reserved space for (currently fixed)
BSMeasurement record[BS_RAWDATA_ARRAYSIZE];	Array of test measurement data
int iTotalNumRecs;	Number of populated records in the array
int ulFirstTestNum;	Test number of the first record stored in the array.

Remarks

Most models index 1000 records and store the last 100 at any given time.

1.1.1.2.9 Bodystat::BSResults Structure

BSResults: Bodystat Results structure. Holds the calculated results for any given test measurement.

C++

```
struct BSResults {
  float fFatPerc;
  float fFatKg;
  float fLeanPerc;
  float fLeanKg;
  float fTotalWeight;
  float fDryLW;
  float fTBWPerc;
  float fTBW;
  float fECWPerc;
  float fECW;
  float fICWPerc;
  float fICW;
  float fBCM;
  float fThirdSpace;
  float fNutrition;
  float fIllness;
  float fBMR;
  float fBMRkg;
  float fEstAvg;
  float fBMI;
  float fBFMI;
  float fFFMI;
  float fWaistHip;
  float fWellness;
  float fECW_Legacy;
  float fTBW_Legacy;
  float fOHY;
  float fSkMuscle;
  float fCm;
```

```
    float fRext;
    float fRint;
    float fFc;
    float fAlpha;
  };
```

C#

```
  public struct BSResults {
    public float fFatPerc;
    public float fFatKg;
    public float fLeanPerc;
    public float fLeanKg;
    public float fTotalWeight;
    public float fDryLW;
    public float fTBWPerc;
    public float fTBW;
    public float fECWPerc;
    public float fECW;
    public float fICWPerc;
    public float fICW;
    public float fBCM;
    public float fThirdSpace;
    public float fNutrition;
    public float fIllness;
    public float fBMR;
    public float fBMRkg;
    public float fEstAvg;
    public float fBMI;
    public float fBFMI;
    public float fFFMI;
    public float fWaistHip;
    public float fWellness;
    public float fECW_Legacy;
    public float fTBW_Legacy;
    public float fOHY;
    public float fSkMuscle;
    public float fCm;
    public float fRext;
    public float fRint;
    public float fFc;
    public float fAlpha;
  }
```

Visual Basic

```
  Public Structure BSResults
    Public fFatPerc As float
    Public fFatKg As float
    Public fLeanPerc As float
    Public fLeanKg As float
    Public fTotalWeight As float
    Public fDryLW As float
    Public fTBWPerc As float
    Public fTBW As float
    Public fECWPerc As float
    Public fECW As float
    Public fICWPerc As float
    Public fICW As float
    Public fBCM As float
    Public fThirdSpace As float
    Public fNutrition As float
    Public fIllness As float
    Public fBMR As float
    Public fBMRkg As float
    Public fEstAvg As float
    Public fBMI As float
    Public fBFMI As float
    Public fFFMI As float
    Public fWaistHip As float
    Public fWellness As float
    Public fECW_Legacy As float
    Public fTBW_Legacy As float
    Public fOHY As float
    Public fSkMuscle As float
    Public fCm As float
    Public fRext As float
    Public fRint As float
    Public fFc As float
    Public fAlpha As float
  End Structure
```

Java

```
public class BSResults;
```

MATLAB

```
BSResults
```

File

BodystatSDK.h (see page 55)

Members

Members	Description
float fFatPerc;	Fat percentage
float fFatKg;	Fat (in kg)
float fLeanPerc;	Lean percentage
float fLeanKg;	Lean (in kg)
float fTotalWeight;	Total Weight (in kg)
float fDryLW;	Dry Lean Weight (in kg)
float fTBWPerc;	Water percentage or Total Body Water percentage
float fTBW;	Water (in litres) or Total Body Water (in litres)
float fECWPerc;	Extra Cellular Water percentage (MDD/Quad only)
float fECW;	Extra Cellular Water (in litres) (MDD/Quad only)
float fICWPerc;	Intra Cellular Water percentage (MDD/Quad only)
float fICW;	Intra Cellular Water (in litres) (MDD/Quad only)
float fBCM;	Body cell mass (Quad only)
float fThirdSpace;	3rd space water (in litres) (Quad only)
float fNutrition;	Nutrition index (Quad only)
float fIllness;	Prediction Marker (TM) - formerly known as Illness Marker (TM) (Quad only)
float fBMR;	Basal Metabolic Rate (in kcal)
float fBMRkg;	Basal Metabolic Rate per kilogram in (kcal/kg)
float fEstAvg;	Estimated Average Requirement (in kcal)
float fBMI;	Body Mass Index
float fBFMI;	Body Fat Mass Index (MDD/Quad only)
float fFFMI;	Fat Free Mass Index (MDD/Quad only)
float fWaistHip;	Waist/Hip ratio
float fWellness;	Wellness Marker (TM) (MDD only)
float fECW_Legacy;	Legacy ECW calculation (QuanScan mode)
float fOHY;	Over hydration
float fSkMuscle;	Skeletal muscle mass
float fCm;	Cell membrane capacitance
float fRext;	R extracellular
float fRint;	R intracellular
float fFc;	Characteristic frequency
float fAlpha;	Alpha angle

1.1.1.2.10 Bodystat::ESMeasurement Structure

ESMeasurement: Equistat Measurement structure. Each test measurement recorded in the device is stored one of these structures. Holds the user inputted information about the horse, together with bio-impedance readings measured by the device (raw electrical readings only, not full results).

C++

```
struct ESMeasurement {
  BSDeviceModel iDeviceModel;
  ULONG ulDeviceSerialNumber;
  __time64_t tTestDate;
  int iHeight;
  int iBCScore;
  int iZ_5kHz;
  int iZ_16kHz;
  int iZ_24kHz;
  int iZ_50kHz;
  int iZ_140kHz;
  int iZ_200kHz;
  int iZ_280kHz;
  float fPA_50kHz;
};
```

C#

```
public struct ESMeasurement {
  public BSDeviceModel iDeviceModel;
  public ULONG ulDeviceSerialNumber;
  public __time64_t tTestDate;
  public int iHeight;
  public int iBCScore;
  public int iZ_5kHz;
  public int iZ_16kHz;
  public int iZ_24kHz;
  public int iZ_50kHz;
  public int iZ_140kHz;
  public int iZ_200kHz;
  public int iZ_280kHz;
  public float fPA_50kHz;
}
```

Visual Basic

```
Public Structure ESMeasurement
  Public iDeviceModel As BSDeviceModel
  Public ulDeviceSerialNumber As ULONG
  Public tTestDate As __time64_t
  Public iHeight As Integer
  Public iBCScore As Integer
  Public iZ_5kHz As Integer
  Public iZ_16kHz As Integer
  Public iZ_24kHz As Integer
  Public iZ_50kHz As Integer
  Public iZ_140kHz As Integer
  Public iZ_200kHz As Integer
  Public iZ_280kHz As Integer
  Public fPA_50kHz As float
End Structure
```

Java

```
public class ESMeasurement;
```

MATLAB

```
ESMeasurement
```

File

BodystatSDK.h (see page 55)

1

Members

Members	Description
BSDeviceModel iDeviceModel;	Device model which performed the measurement
ULONG ulDeviceSerialNumber;	Unique serial number of the device which performed the measurement
__time64_t tTestDate;	Date & time the test measurement was performed (Windows 64-bit time structure) (ES Pro only)
int iHeight;	Inputted height of the subject (centimeters)
int iBCScore;	Inputted BC score
int iZ_5kHz;	Impedance measured at 5 kHz (result in ohms)
int iZ_16kHz;	Impedance measured at 16 kHz (result in ohms)
int iZ_24kHz;	Impedance measured at 24 kHz (result in ohms)
int iZ_50kHz;	Impedance measured at 50 kHz (result in ohms)
int iZ_140kHz;	Impedance measured at 140 kHz (result in ohms)
int iZ_200kHz;	Impedance measured at 200 kHz (result in ohms)
int iZ_280kHz;	Impedance measured at 280 kHz (result in ohms)
float fPA_50kHz;	Phase angle measured at 50 kHz (result in degrees) (ES Pro only)

1.1.1.2.11 Bodystat::ESNormals Structure

ESNormals: Equistat Normals structure. Holds the normal values or normal ranges for any given test measurement.

C++

```
struct ESNormals {
  int iFatKg_L;
  int iFatKg_H;
  int iFatKg_PBMPerc_L;
  int iFatKg_PBMPerc_H;
  int iTFV_L;
  int iTFV_H;
  int iTFVPerc_L;
  int iTFVPerc_H;
  int iECFV_L;
  int iECFV_H;
  int iECFV_PBM_L;
  int iECFV_PBM_H;
  int iECFV_TFVPerc_L;
  int iECFV_TFVPerc_H;
  int iICFV_L;
  int iICFV_H;
  int iICFV_PBM_L;
  int iICFV_PBM_H;
  int iICFV_TFVPerc_L;
  int iICFV_TFVPerc_H;
  int iPV_L;
  int iPV_H;
  int iPV_ECFVPerc_L;
  int iPV_ECFVPerc_H;
};
```

C#

```
public struct ESNormals {
  public int iFatKg_L;
  public int iFatKg_H;
  public int iFatKg_PBMPerc_L;
  public int iFatKg_PBMPerc_H;
  public int iTFV_L;
  public int iTFV_H;
  public int iTFVPerc_L;
  public int iTFVPerc_H;
```

```
    public int iECFV_L;
    public int iECFV_H;
    public int iECFV_PBM_L;
    public int iECFV_PBM_H;
    public int iECFV_TFVPerc_L;
    public int iECFV_TFVPerc_H;
    public int iICFV_L;
    public int iICFV_H;
    public int iICFV_PBM_L;
    public int iICFV_PBM_H;
    public int iICFV_TFVPerc_L;
    public int iICFV_TFVPerc_H;
    public int iPV_L;
    public int iPV_H;
    public int iPV_ECFVPerc_L;
    public int iPV_ECFVPerc_H;
  }
```

Visual Basic

```
  Public Structure ESNormals
    Public iFatKg_L As Integer
    Public iFatKg_H As Integer
    Public iFatKg_PBMPerc_L As Integer
    Public iFatKg_PBMPerc_H As Integer
    Public iTFV_L As Integer
    Public iTFV_H As Integer
    Public iTFVPerc_L As Integer
    Public iTFVPerc_H As Integer
    Public iECFV_L As Integer
    Public iECFV_H As Integer
    Public iECFV_PBM_L As Integer
    Public iECFV_PBM_H As Integer
    Public iECFV_TFVPerc_L As Integer
    Public iECFV_TFVPerc_H As Integer
    Public iICFV_L As Integer
    Public iICFV_H As Integer
    Public iICFV_PBM_L As Integer
    Public iICFV_PBM_H As Integer
    Public iICFV_TFVPerc_L As Integer
    Public iICFV_TFVPerc_H As Integer
    Public iPV_L As Integer
    Public iPV_H As Integer
    Public iPV_ECFVPerc_L As Integer
    Public iPV_ECFVPerc_H As Integer
  End Structure
```

Java

```
  public class ESNormals;
```

MATLAB

```
  ESNormals
```

File

BodystatSDK.h (see page 55)

Members

Members	Description
int iFatKg_L;	Normal range for fat (in kg) - (low end)
int iFatKg_H;	Normal range for fat (in kg) - (high end)
int iFatKg_PBMPerc_L;	Normal range for fat kg pbm percentage - (low end)
int iFatKg_PBMPerc_H;	Normal range for fat kg pbm percentage - (high end)
int iTFV_L;	Normal range for TFV - (low end)
int iTFV_H;	Normal range for TFV - (high end)
int iTFVPerc_L;	Normal range for TFV percentage - (low end)
int iTFVPerc_H;	Normal range for TFV percentage - (high end)

int iECFV_L;	Normal range for ECFV - (low end)
int iECFV_H;	Normal range for ECFV - (high end)
int iECFV_PBM_L;	Normal range for ECFV PBM - (low end)
int iECFV_PBM_H;	Normal range for ECFV PBM - (high end)
int iECFV_TFVPerc_L;	Normal range for ECFV TFV percentage - (low end)
int iECFV_TFVPerc_H;	Normal range for ECFV TFV percentage - (high end)
int iICFV_L;	Normal range for ICFV - (low end)
int iICFV_H;	Normal range for ICFV - (high end)
int iICFV_PBM_L;	Normal range for ICFV PBM - (low end)
int iICFV_PBM_H;	Normal range for ICFV PBM - (high end)
int iICFV_TFVPerc_L;	Normal range for ICFV TFV percentage - (low end)
int iICFV_TFVPerc_H;	Normal range for ICFV TFV percentage - (high end)
int iPV_L;	Normal range for PV - (low end)
int iPV_H;	Normal range for PV - (high end)
int iPV_ECFVPerc_L;	Normal range for PV ECFV percentage - (low end)
int iPV_ECFVPerc_H;	Normal range for PV ECFV percentage - (high end)

1.1.1.2.12 Bodystat::ESRawData Structure

ESRawData: Equistat Raw Data structure. Used when downloading test measurements from the device. Contains a fixed array of measurement records together with information about the number of records occupied within the array and test record starting number.

C++

```
struct ESRawData {
  int iRecordArraySize;
  ESMeasurement record[ES_RAWDATA_ARRAYSIZE];
  int iTotalNumRecs;
  int ulFirstTestNum;
};
```

C#

```
public struct ESRawData {
  public int iRecordArraySize;
  public ESMeasurement record;
  public int iTotalNumRecs;
  public int ulFirstTestNum;
}
```

Visual Basic

```
Public Structure ESRawData
  Public iRecordArraySize As Integer
  Public record As ESMeasurement
  Public iTotalNumRecs As Integer
  Public ulFirstTestNum As Integer
End Structure
```

Java

```
public class ESRawData;
```

MATLAB

```
ESRawData
```

File

BodystatSDK.h (see page 55)

Members

Members	Description
int iRecordArraySize;	Maximum number of records this structure has reserved space for (currently fixed)
ESMeasurement record[ES_RAWDATA_ARRAYSIZE];	Array of test measurement data
int iTotalNumRecs;	Number of populated records in the array
int ulFirstTestNum;	Test number of the first record stored in the array.

Remarks

Most models index 1000 records and store the last 100 at any given time.

1.1.1.2.13 Bodystat::ESResults Structure

ESResults: Equistat Results structure. Holds the calculated results for any given test measurement.

C++

```
struct ESResults {
  float fFatKg;
  float fPBM;
  float fIBM;
  float fFatKg_PBMPerc;
  float fTFV;
  float fTFVPerc;
  float fECFV;
  float fECFV_PBM;
  float fECFV_TFVPerc;
  float fICFV;
  float fICFV_PBM;
  float fICFV_TFVPerc;
  float fPV;
  float fPV_ECFVPerc;
  float fDehydration;
};
```

C#

```
public struct ESResults {
  public float fFatKg;
  public float fPBM;
  public float fIBM;
  public float fFatKg_PBMPerc;
  public float fTFV;
  public float fTFVPerc;
  public float fECFV;
  public float fECFV_PBM;
  public float fECFV_TFVPerc;
  public float fICFV;
  public float fICFV_PBM;
  public float fICFV_TFVPerc;
  public float fPV;
  public float fPV_ECFVPerc;
  public float fDehydration;
}
```

Visual Basic

```
Public Structure ESResults
  Public fFatKg As float
  Public fPBM As float
  Public fIBM As float
  Public fFatKg_PBMPerc As float
  Public fTFV As float
  Public fTFVPerc As float
  Public fECFV As float
  Public fECFV_PBM As float
  Public fECFV_TFVPerc As float
```

```
    Public fICFV As float
    Public fICFV_PBM As float
    Public fICFV_TFVPerc As float
    Public fPV As float
    Public fPV_ECFVPerc As float
    Public fDehydration As float
  End Structure
```

Java

```
  public class ESResults;
```

MATLAB

```
  ESResults
```

File

BodystatSDK.h (see page 55)

Members

Members	Description
float fFatKg;	Fat (in kg)
float fPBM;	PBM
float fIBM;	IBM
float fFatKg_PBMPerc;	Fat Kg PBM percentage
float fTFV;	TFV
float fTFVPerc;	TFV Percentage
float fECFV;	ECFV
float fECFV_PBM;	ECFV PBM
float fECFV_TFVPerc;	ECFV TFV percentage
float fICFV;	ICFV
float fICFV_PBM;	ICFV PBM
float fICFV_TFVPerc;	ICFV TFV percentage
float fPV;	PV
float fPV_ECFVPerc;	PV ECFV percentage
float fDehydration;	Dehydration

1.1.1.2.14 ImpData Structure

Represents one frequency measurement

C++

```
  struct ImpData {
    float fFrequency;
    float fImpedance;
    float fPhaseAngle;
  };
```

C#

```
  public struct ImpData {
    public float fFrequency;
    public float fImpedance;
    public float fPhaseAngle;
  }
```

Visual Basic

```
  Public Structure ImpData
    Public fFrequency As float
    Public fImpedance As float
    Public fPhaseAngle As float
  End Structure
```

Java

```
public class ImpData;
```

MATLAB

```
ImpData
```

File

BodystatSDK.h (see page 55)

ImpData Data Members

	Name	Description
	fFrequency (see page 50)	Frequency in kHz
	fImpedance (see page 50)	Absolute value of impedance measured at given frequency (result in ohms)
	fPhaseAngle (see page 51)	Phase angle (result in degrees)

ImpData Methods

	Name	Description
	ImZ (see page 51)	Imaginary part of impedance, also reactance Im Z = |Z|*sin(Phase)
	ReZ (see page 51)	Real part of impedance, also resistance Re Z = |Z|*cos(Phase)

1.1.1.2.14.1 ImpData Data Members

1.1.1.2.14.1.1 ImpData::fFrequency Data Member

C++

```
float fFrequency;
```

C#

```
public float fFrequency;
```

Visual Basic

```
Public fFrequency As float
```

Java

```
public float fFrequency;
```

MATLAB

```
fFrequency
```

Description

Frequency in kHz

1.1.1.2.14.1.2 ImpData::fImpedance Data Member

C++

```
float fImpedance;
```

C#

```
public float fImpedance;
```

Visual Basic

```
Public fImpedance As float
```

Java

```
public float fImpedance;
```

MATLAB

```
fImpedance
```

Description

Absolute value of impedance measured at given frequency (result in ohms)

1.1.1.2.14.1.3 ImpData::fPhaseAngle Data Member

C++

```
float fPhaseAngle;
```

C#

```
public float fPhaseAngle;
```

Visual Basic

```
Public fPhaseAngle As float
```

Java

```
public float fPhaseAngle;
```

MATLAB

```
fPhaseAngle
```

Description

Phase angle (result in degrees)

1.1.1.2.14.2 ImpData Methods

1.1.1.2.14.2.1 ImpData::ImZ Method

Imaginary part of impedance, also reactance Im Z = |Z|*sin(Phase)

C++

```
float ImZ() const;
```

C#

```
public float ImZ();
```

Visual Basic

```
Public Function ImZ() As float
```

Java

```
public float ImZ();
```

MATLAB

```
ImZ
```

Body Source

```
float ImZ() const
{
 return fImpedance * sin(fPhaseAngle * M_PI / 180);
}
```

1.1.1.2.14.2.2 ImpData::ReZ Method

Real part of impedance, also resistance Re Z = |Z|*cos(Phase)

C++

```
float ReZ() const;
```

C#

```
public float ReZ();
```

Visual Basic

```
Public Function ReZ() As float
```

Java

```
public float ReZ();
```

MATLAB

```
ReZ
```

Body Source

```
float ReZ() const
{
 return fImpedance * cos(fPhaseAngle * M_PI / 180);
}
```

1.1.1.3 Variables

The following table lists variables in this documentation.

Variables

Name	Description
BS_RAWDATA_ARRAYSIZE (see page 52)	Number of measurement records BSRawData (see page 40) structure should reserved space for (currently fixed)
ES_RAWDATA_ARRAYSIZE (see page 52)	Number of measurement records ESRawData (see page 47) structure should reserved space for (currently fixed)

1.1.1.3.1 Bodystat::BS_RAWDATA_ARRAYSIZE Variable

C++

```
const int BS_RAWDATA_ARRAYSIZE = 100;
```

C#

```
BS_RAWDATA_ARRAYSIZE
```

Visual Basic

```
BS_RAWDATA_ARRAYSIZE
```

Java

```
BS_RAWDATA_ARRAYSIZE
```

MATLAB

```
BS_RAWDATA_ARRAYSIZE
```

File

BodystatSDK.h (see page 55)

Description

Number of measurement records BSRawData (see page 40) structure should reserved space for (currently fixed)

1.1.1.3.2 Bodystat::ES_RAWDATA_ARRAYSIZE Variable

C++

```
const int ES_RAWDATA_ARRAYSIZE = 100;
```

C#

```
ES_RAWDATA_ARRAYSIZE
```

Visual Basic

```
ES_RAWDATA_ARRAYSIZE
```

Java

```
ES_RAWDATA_ARRAYSIZE
```

MATLAB

```
ES_RAWDATA_ARRAYSIZE
```

File

BodystatSDK.h (see page 55)

Description

Number of measurement records ESRawData (see page 47) structure should reserved space for (currently fixed)

1.1.2 Macros

The following table lists macros in this documentation.

Macros

Name	Description
BODYSTATSDK_API (see page 53)	The following ifdef block is the standard way of creating macros which make exporting from a DLL simpler. All files within this DLL are compiled with the BODYSTATSDK_EXPORTS symbol defined on the command line. This symbol should not be defined on any project that uses this DLL. This way any other project whose source files include this file see BODYSTATSDK_API functions as being imported from a DLL, whereas this DLL sees symbols defined with this macro as being exported.
M_PI (see page 54)	This is macro M_PI.

Module

Bodystat API - Main (see page 1)

1.1.2.1 BODYSTATSDK_API Macro

C++

```
#define BODYSTATSDK_API extern "C" __declspec(dllimport)
```

C#

```
#define BODYSTATSDK_API extern "C" __declspec(dllimport)
```

Visual Basic

```
#Const BODYSTATSDK_API = extern "C" __declspec(dllimport)
```

Java

```
BODYSTATSDK_API
```

MATLAB

```
BODYSTATSDK_API
```

File

BodystatSDK.h (see page 55)

Description

The following ifdef block is the standard way of creating macros which make exporting from a DLL simpler. All files within this DLL are compiled with the BODYSTATSDK_EXPORTS symbol defined on the command line. This symbol should not be defined on any project that uses this DLL. This way any other project whose source files include this file see BODYSTATSDK_API functions as being imported from a DLL, whereas this DLL sees symbols defined with this macro as being exported.

1.1.2.2 M_PI Macro

C++

```
#define M_PI 3.14159265358979323846
```

C#

```
#define M_PI 3.14159265358979323846
```

Visual Basic

```
#Const M_PI = 3.14159265358979323846
```

Java

```
M_PI
```

MATLAB

```
M_PI
```

File

BodystatSDK.h (see page 55)

Description

This is macro M_PI.

1.1.3 Files

The following table lists files in this documentation.

Files

Name	Description
BodystatSDK.h (see page 55)	Bodystat SDK (C) 2010-2014 Bodystat (Isle of Man) Ltd. All rights reserved. Use subject to license. Web: http://www.bodystat.com Email: info@bodystat.com Tel: +44 (0)1624 629 571 Fax: +44 (0)1624 611 544 Version 1.36 (09/May/2014) Selected download is now passed through to the results form in the C# sample. Fixed error in VB.Net wrapper (preventing download working) in the VB.Net sample. Fixed some missing normals for MST device. 1.35 (10/Mar/2014) Added Cole methods for ECW/ICW calculations used in the MST device. Fixed a memory pointer issue for the detected bodystat device info structure. 1.34 (12/Nov/2013) Exposed Bluetooth API search timeout values to... more (see page 55)

Module

Bodystat API - Main (see page 1)

1.1.3.1 BodystatSDK.h

Bodystat SDK

Web: http://www.bodystat.com

Email: info@bodystat.com

Tel: +44 (0)1624 629 571

Fax: +44 (0)1624 611 544

Version

1.36 (09/May/2014)

Selected download is now passed through to the results form in the C# sample.

Fixed error in VB.Net wrapper (preventing download working) in the VB.Net sample.

Fixed some missing normals for MST device.

1.35 (10/Mar/2014)

Added Cole methods for ECW/ICW calculations used in the MST device.

Fixed a memory pointer issue for the detected bodystat device info structure.

1.34 (12/Nov/2013)

Exposed Bluetooth API search timeout values to the SDK

1.33 (28/Sep/2012)

Removed overly aggressive constraints on measurement parameters in CalculateResults and CalculateNormals()

Added ECW and ICW to MDD results.

Updated MFC/VB/C# samples to support ECW and ICW for MDD results.

Renamed Illness Maker(TM) to Prediction Marker(TM) - naming/labeling change only, code left unchanged to avoid breaking change.

1.32 (08/Aug/2011)

Corrected a bug in which GetBodystatDevice could erroneously return success with an invalid device even though no device was found

Corrected message box title when pairing new devices

Updated C# wrapper with get/set accessors for databinding [Breaking Change: Minor refactoring will be required to accommodate new accessor names]

Updated C# wrapper with additional wrappers for calls with StringBuilder parameters (now callable directly with string parameters)

Updated C# wrapper to force imports as Cdecl types

Fixed bug in C# wrapper causing invalid parameter error during download

Updated VB wrapper to force imports as Cdecl types

Updated C# WinForms sample to use new string wrappers

1.31 (26/Oct/2010)

Added C# wrapper and C# WinForms sample

Added initial support for Equistat devices

1.30 (29/Aug/2010)

Added VB.Net wrapper and VB WinForms sample

1.29 (30/Jun/2010)

Initial Release

Macros

Name	Description
BODYSTATSDK_API (see page 53)	The following ifdef block is the standard way of creating macros which make exporting from a DLL simpler. All files within this DLL are compiled with the BODYSTATSDK_EXPORTS symbol defined on the command line. This symbol should not be defined on any project that uses this DLL. This way any other project whose source files include this file see BODYSTATSDK_API functions as being imported from a DLL, whereas this DLL sees symbols defined with this macro as being exported.
M_PI (see page 54)	This is macro M_PI.

Namespaces

Name	Description
Bodystat (see page 2)	This is the main namespace for the Bodystat API. The entire functions within the Bodystat API are enclosed within the 'Bodystat' namespace.

1.2 C# .NET Wrapper

Bodystat API C# .NET wrapper.

Provides a wrapper interface to the Bodystat DLL with C# calls and types you can work with directly in your C# .NET program. The wrapper handles the necessary marshalling and interop issues required to interface with the native unmanaged Bodystat DLL.

Namespaces

Name	Description
BodystatAPI (see page 57)	This is the Bodystat API namespace for our Bodystat DLL wrapper. See main BodystatAPI class members for more information.

Files

Name	Description
BodystatSDK.cs (see page 80)	Bodystat SDK - C# Wrapper (C) 2010-2014 Bodystat (Isle of Man) Ltd. All rights reserved. Use subject to license. Web: http://www.bodystat.com Email: info@bodystat.com Tel: +44 (0)1624 629 571 Fax: +44 (0)1624 611 544 Version For version information please refer to main BodystatSDK API

1.2.1 Bodystat API

This is the Bodystat API namespace for our Bodystat DLL wrapper.

See main BodystatAPI class members for more information.

Description

Main documentation for the functions, definitions, classes and structures provided by the Bodystat API.

Module

C# .NET Wrapper (see page 56)

1.2.1.1 BodystatAPI.BodystatAPI.BSAutoSetupBT@out string@bool@IntPtr

Automatic Bluetooth setup. Automatically searches available Bluetooth devices, exchanges passkeys with Bodystat device and enables serial port service required for communication with the device. If multiple Bodystat devices are detected the user is prompted to chose the desired device (model and BT BDA address shown).

This function is typically called as a one time setup for the device (e.g. after first installation, or from the hardware configuration page of your application's settings, or when the user obtains a replacement device (upgrades/replacement/etc).

Typically, once setup you need only store the device com port returned. You can then subsequently communicate with the device knowing only the com port. It is not necessary to setup the device in this way before each use.

New non-paired units will only be found if switched on, in range and ready for communication (on main startup screen). It is advisable to instruct the user to ensure this is the case before calling.

Parameters

Parameters	Description
lpComPort	[out] String buffer to receive the device port the device is assigned to.
bReportErrors	TRUE to report errors to the user (shown as message boxes, locale constraints apply). FALSE to hide errors from the user (still not completely silent).
hWndParent	Handle of the parent window.

Returns

TRUE if it succeeds, FALSE if it fails.

Remarks

Note this process can be time consuming to complete and executes synchronously. Either call from a threaded function or illustrate operation in progress to the user e.g. by means of an indefinite progress bar or busy mouse cursor, until the task completes or times out.

Errors during the process can be optionally shown/hidden. However, a completely silent approach is not possible using this function (instead use BSSearchBTDevices and BSGetBTBodystatDevice or iterate radio handles manually. Then pass desired handle to BSAuthenticateBTDevice for pairing).

Finally be aware this function is only available for supported locales. Otherwise English messages may be displayed to the user. Adopt silent approach discussed previously if you are targeting a non-supported locale.

1.2.1.2 BodystatAPI.BodystatAPI.BSAutoSetupBT@StringBuilder@int@bool@IntPtr

Automatic Bluetooth setup. Automatically searches available Bluetooth devices, exchanges passkeys with Bodystat device and enables serial port service required for communication with the device. If multiple Bodystat devices are detected the user is prompted to chose the desired device (model and BT BDA address shown).

This function is typically called as a one time setup for the device (e.g. after first installation, or from the hardware configuration page of your application's settings, or when the user obtains a replacement device (upgrades/replacement/etc).

Typically, once setup you need only store the device com port returned. You can then subsequently communicate with the device knowing only the com port. It is not necessary to setup the device in this way before each use.

New non-paired units will only be found if switched on, in range and ready for communication (on main startup screen). It is advisable to instruct the user to ensure this is the case before calling.

Parameters

Parameters	Description
lpComPort	[out] String buffer to receive the device port the device is assigned to.
iSize	The size of the buffer.
bReportErrors	TRUE to report errors to the user (shown as message boxes, locale constraints apply). FALSE to hide errors from the user (still not completely silent).
hWndParent	Handle of the parent window.

Returns

TRUE if it succeeds, FALSE if it fails.

Remarks

Note this process can be time consuming to complete and executes synchronously. Either call from a threaded function or illustrate operation in progress to the user e.g. by means of an indefinite progress bar or busy mouse cursor, until the task completes or times out.

Errors during the process can be optionally shown/hidden. However, a completely silent approach is not possible using this function (instead use BSSearchBTDevices and BSGetBTBodystatDevice or iterate radio handles manually. Then pass desired handle to BSAuthenticateBTDevice for pairing).

Finally be aware this function is only available for supported locales. Otherwise English messages may be displayed to the user. Adopt silent approach discussed previously if you are targeting a non-supported locale.

1.2.1.3 BodystatAPI.BodystatAPI.BSCalculateNormals@BSMeasurement@BSResults@out BSNormals

Calculate normal values and normal ranges for a given subject and their bio-impedance measurement together with the matching results structure for that measurement. You must supply a populated results structure which pertains to the measurement being supplied (obtained from BSCalculateResults).

Parameters

Parameters	Description
pM	The measurement details downloaded from the device (includes model, user inputted parameters and bio-impedance measurement readings).
pR	The results calculated for this measurement previously.
pN	[out] Receives the normal values and normal ranges for this measurement.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

Normals vary by subject attributes, measured bio-impedance values and calculated results.

1.2.1.4 BodystatAPI.BodystatAPI.BSCalculateResults@BSMeasurement@out BSResults@BSAnalysisMode

Calculate complete result set for a given bio-impedance measurement recorded by the Bodystat device.

Parameters

Parameters	Description
pM	The measurement details downloaded from the device (includes model, user inputted parameters and bio-impedance measurement readings).
pR	[out] Receives the complete calculated results for this measurement.
iAnalysisMode	The desired analysis mode (see BSAnalysisMode for more information).

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

Results vary by subject attributes and measured bio-impedance values.

1.2.1.5 BodystatAPI.BodystatAPI.BSCloseComport

Close com port.

1.2.1.6 BodystatAPI.BodystatAPI.BSConnect

Connect to the Bodystat device and ensure the device is responding.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have opened the com port by calling BSOpenComPort previously. This connect command must be issued before sending other any other commands.

1.2.1.7 BodystatAPI.BodystatAPI.BSGetBTStackInfo@out string

Retrieve information about the Bluetooth stack. Useful for diagnostics.

Parameters

Parameters	Description
lpBuffer	[out] String buffer to receive the information.

Returns

true if it succeeds, false if it fails.

1.2.1.8 BodystatAPI.BodystatAPI.BSGetBTStackInfo@StringBuilder@int

Retrieve information about the Bluetooth stack. Useful for diagnostics.

Parameters

Parameters	Description
lpBuffer	[out] String buffer to receive the information.
iBufferSize	Size of the buffer.

Returns

true if it succeeds, false if it fails.

Remarks

Note: Information can only be provided for supported BT radios.

1.2.1.9 BodystatAPI.BodystatAPI.BSGetDeviceModelName@BSDeviceModel@out string

Determine the full product name of a specific model.

Parameters

Parameters	Description
iModel	The desired model of interest.
lpBuffer	[out] String buffer to receive the name.
iBufferSize	Size of the buffer.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

1.2.1.10 BodystatAPI.BodystatAPI.BSGetDeviceModelName@BSDeviceModel@StringBuilder@int

Determine the full product name of a specific model.

Parameters

Parameters	Description
iModel	The desired model of interest.
lpBuffer	[out] String buffer to receive the name.
iBufferSize	Size of the buffer.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

1.2.1.11 BodystatAPI.BodystatAPI.BSGetDeviceModelNameShort@BSDeviceModel@out string

Determine the short product name of a specific model.

Parameters

Parameters	Description
iModel	The desired model of interest.
lpBuffer	[out] String buffer to receive the name.
iBufferSize	Size of the buffer.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

1.2.1.12 BodystatAPI.BodystatAPI.BSGetDeviceModelNameShort@BSDeviceModel@StringBuilder@int

Determine the short product name of a specific model.

Parameters

Parameters	Description
iModel	The desired model of interest.
lpBuffer	[out] String buffer to receive the name.
iBufferSize	Size of the buffer.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

1.2.1.13 BodystatAPI.BodystatAPI.BSGetSdkLibraryVersion@out int@out int

Determine the Bodystat SDK library (DLL) version.

Parameters

Parameters	Description
pMajorSdkVer	[out] Receives the major version number of the Bodystat SDK DLL.
pMinorSdkVer	[out] Receives the minor version number of the Bodystat SDK DLL.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

Caller should check the version number of the DLL if utilising functionality noted only as being available in later revisions of the SDK. Or, alternatively, ensure the SDK DLL is updated appropriately during installation of your application.

1.2.1.14 BodystatAPI.BodystatAPI.BSIsBTAvailable

Check if a supported BT radio is available on this computer. Checks for presence of USB Bluetooth dongles and inbuilt Bluetooth modules in the PC. No communication with Bodystat devices takes place during this query.

Returns

Returns true if supported radio is available and enabled.

Remarks

Note: Only supported BT radios can be queried (presently those compatible with the Microsoft Bluetooth APIs and some Widdcomm models).

Also note, Bluetooth must be enabled on the PC by the user to permit detection. Some PCs, laptops particularly, have physical switches to enable bluetooth/wireless communication.

1.2.1.15 BodystatAPI.BodystatAPI.BSNormals.AerobicCapacity_H

High for Average grade in Aerobic Capacity normals (mlsO2/kg/min)

1.2.1.16 BodystatAPI.BodystatAPI.BSNormals.AerobicCapacity_L

Low for Average grade in Aerobic Capacity normals (mlsO2/kg/min)

1.2.1.17 BodystatAPI.BodystatAPI.BSNormals.BFMI_H

Normal range for Body Fat Mass Index (MDD/Quad only) - (high end)

1.2.1.18 BodystatAPI.BodystatAPI.BSNormals.BFMI_L

Normal range for Body Fat Mass Index (MDD/Quad only) - (low end)

1.2.1.19 BodystatAPI.BodystatAPI.BSNormals.BMI_H

Normal range for Body Mass Index - (high end)

1.2.1.20 BodystatAPI.BodystatAPI.BSNormals.BMI_L

Normal range for Body Mass Index - (low end)

1.2.1.21 BodystatAPI.BodystatAPI.BSNormals.Cholesterol_HDL_H

Cholesterol HDL high range in mg/dl

1.2.1.22 BodystatAPI.BodystatAPI.BSNormals.Cholesterol_HDL_L

Cholesterol HDL low range in mg/dl

1.2.1.23 BodystatAPI.BodystatAPI.BSNormals.Cholesterol_LDL_H

Cholesterol VDL high range in mg/dl

1.2.1.24 BodystatAPI.BodystatAPI.BSNormals.Cholesterol_LDL_L

Cholesterol VDL low range in mg/dl

1.2.1.25 BodystatAPI.BodystatAPI.BSNormals.Cholesterol_Total_H

Cholesterol total high range in mg/dl

1.2.1.26 BodystatAPI.BodystatAPI.BSNormals.Cholesterol_Total_L

Cholesterol total low range in mg/dl

1.2.1.27 BodystatAPI.BodystatAPI.BSNormals.Cholesterol_Triglycerides_H

Triglycerides high range in mg/dl

1.2.1.28 BodystatAPI.BodystatAPI.BSNormals.Cholesterol_Triglycerides_L

Triglycerides low range in mg/dl

1.2.1.29 BodystatAPI.BodystatAPI.BSNormals.Cholesterol_VLDL_H

Cholesterol VLDL high range in mg/dl

1.2.1.30 BodystatAPI.BodystatAPI.BSNormals.Cholesterol_VLDL_L

Cholesterol VLDL low range in mg/dl

1.2.1.31 BodystatAPI.BodystatAPI.BSNormals.ECWPerc_Norm

Normal value for ECW Percentage (MDD/Quad only)

1.2.1.32 BodystatAPI.BodystatAPI.BSNormals.FatKg_H

Normal range for fat (in kg) - (high end)

1.2.1.33 BodystatAPI.BodystatAPI.BSNormals.FatKg_L

Normal range for fat (in kg) - (low end)

1.2.1.34 BodystatAPI.BodystatAPI.BSNormals.FatPerc_H

Normal range for fat percentage - (high end)

1.2.1.35 BodystatAPI.BodystatAPI.BSNormals.FatPerc_L

Normal range for fat percentage - (low end)

1.2.1.36 BodystatAPI.BodystatAPI.BSNormals.FFMI_H

Normal range for Fat Free Mass Index (MDD/Quad only) - (high end)

1.2.1.37 BodystatAPI.BodystatAPI.BSNormals.FFMI_L

Normal range for Fat Free Mass Index (MDD/Quad only) - (low end)

1.2.1.38 BodystatAPI.BodystatAPI.BSNormals.Flexibility_H

Flexibility high range of normal in cm

1.2.1.39 BodystatAPI.BodystatAPI.BSNormals.Flexibility_L

Flexibility low range of normal in cm

1.2.1.40 BodystatAPI.BodystatAPI.BSNormals.GlucoseFasting_H

Glucose (mg/dL)

1.2.1.41 BodystatAPI.BodystatAPI.BSNormals.GlucoseFasting_L

Glucose (mg/dL)

1.2.1.42 BodystatAPI.BodystatAPI.BSNormals.GlucoseNonFasting_H

Glucose (mg/dL)

1.2.1.43 BodystatAPI.BodystatAPI.BSNormals.GlucoseNonFasting_L

Glucose (mg/dL)

1.2.1.44 BodystatAPI.BodystatAPI.BSNormals.GripLeft

Grip left normal value in kg

1.2.1.45 BodystatAPI.BodystatAPI.BSNormals.GripRight

Grip right normal value in kg

1.2.1.46 BodystatAPI.BodystatAPI.BSNormals.ICWPerc_Norm

Normal value for ICW Percentage (MDD/Quad only)

1.2.1.47 BodystatAPI.BodystatAPI.BSNormals.Illness_H

Normal range for Illness Marker (TM) (Quad only) - (high end)

1.2.1.48 BodystatAPI.BodystatAPI.BSNormals.Illness_L

Normal range for Illness Marker (TM) (Quad only) - (low end)

1.2.1.49 BodystatAPI.BodystatAPI.BSNormals.LeanKg_H

Normal range for lean (in kg) - (high end)

1.2.1.50 BodystatAPI.BodystatAPI.BSNormals.LeanKg_L

Normal range for lean (in kg) - (low end)

1.2.1.51 BodystatAPI.BodystatAPI.BSNormals.LeanPerc_H

Normal range for lean percentage - (high end)

1.2.1.52 BodystatAPI.BodystatAPI.BSNormals.LeanPerc_L

Normal range for lean percentage - (low end)

1.2.1.53 BodystatAPI.BodystatAPI.BSNormals.Nutrition_Norm

Normal value for Nutrition Index (Quad only)

1.2.1.54 BodystatAPI.BodystatAPI.BSNormals.TBW_H

Normal range for Total Body Water (in litres) - (high end)

1.2.1.55 BodystatAPI.BodystatAPI.BSNormals.TBW_L

Normal range for Total Body Water (in litres) - (low end)

1.2.1.56 BodystatAPI.BodystatAPI.BSNormals.TBWPerc_H

Normal range for Total Body Water percentage - (high end)

1.2.1.57 BodystatAPI.BodystatAPI.BSNormals.TBWPerc_L

Normal range for Total Body Water percentage - (low end)

1.2.1.58 BodystatAPI.BodystatAPI.BSNormals.TotalWeight_H

Normal range for Total Weight (in kg) - (high end)

1.2.1.59 BodystatAPI.BodystatAPI.BSNormals.TotalWeight_L

Normal range for Total Weight (in kg) - (low end)

1.2.1.60 BodystatAPI.BodystatAPI.BSNormals.TotalWeightMethod

Method used when calculating Total Weight normal range (0 = Composition Method, 1 = BMI Method)

1.2.1.61 BodystatAPI.BodystatAPI.BSNormals.WaistHip_Norm

Normal value for Waist/Hip ratio

1.2.1.62 BodystatAPI.BodystatAPI.BSNormals.Wellness_H

Normal range for Wellness Marker (TM) (MDD only) - (high end)

1.2.1.63 BodystatAPI.BodystatAPI.BSNormals.Wellness_L

Normal range for Wellness Marker (TM) (MDD only) - (low end)

1.2.1.64 BodystatAPI.BodystatAPI.BSOpenComport@string

Open com port to the device ready for communication.

Parameters

Parameters	Description
lpComPort	String buffer containing the device/port to be opened.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

The port must be closed once you are finished sending commands. To save battery power on the device, it is recommended you chain command sequences together and open/close the port immediately before and after the chain sequence.

1.2.1.65 BodystatAPI.BodystatAPI.BSOpenComport@StringBuilder@int

Open com port to the device ready for communication.

Parameters

Parameters	Description
lpComPort	String buffer containing the device/port to be opened.
iSize	The size of the buffer.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

The port must be closed once you are finished sending commands. To save battery power on the device, it is recommended you chain command sequences together and open/close the port immediately before and after the chain sequence.

1.2.1.66 BodystatAPI.BodystatAPI.BSRawData.BSRawData@int

Override the constructor to initialise required variables in the structure. You must ensure you allocate the structure in a manner which calls this function.

Parameters

Parameters	Description
	The parameter is not required by the wrapper and is ignored.
iDontCare	=0

1.2.1.67 BodystatAPI.BodystatAPI.BSReadCalibrationTime@out DateTime

Read the calibration date from the connected device (when the unit was last calibrated). Only the date element of this value should be shown. Exact time value is not relevant.

Parameters

Parameters	Description
dtCalibrationTime	[out] Receives the date the device was last calibrated.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect previously.

1.2.1.68 BodystatAPI.BodystatAPI.BSReadCalibrationTime@out Int64

Read the calibration date from the connected device (when the unit was last calibrated). Only the date element of this value should be shown. Exact time value is not relevant.

Parameters

Parameters	Description
pCalibrationTime	[out] Receives the date the device was last calibrated.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect previously.

1.2.1.69 BodystatAPI.BodystatAPI.BSReadCurrentTime@out DateTime@int

Read the current time from the internal clock of the connected device. Note not all device models support this function (not all have an internal clock).

Parameters

Parameters	Description
dtCurrentTime	[out] Receives the current time of the device's internal clock.
iDST	Daylight Saving Time handling (as per Microsoft time functions) Zero (0) to indicate that standard time is in effect. A value greater than 0 to indicate that daylight saving time is in effect. A value less than zero to have the C run-time library code compute whether standard time or daylight saving time is in effect. (default) = -1

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect previously.

1.2.1.70 BodystatAPI.BodystatAPI.BSReadCurrentTime@out Int64@int

Read the current time from the internal clock of the connected device. Note not all device models support this function (not all have an internal clock).

Parameters

Parameters	Description
pCurrentTime	[out] Receives the current time of the device's internal clock.
iDST	Daylight Saving Time handling (as per Microsoft time functions) Zero (0) to indicate that standard time is in effect. A value greater than 0 to indicate that daylight saving time is in effect. A value less than zero to have the C run-time library code compute whether standard time or daylight saving time is in effect. (default) = -1

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect previously.

1.2.1.71 BodystatAPI.BodystatAPI.BSReadModelVersion@out BSDeviceModel@out byte@out byte@out byte@out byte@out byte

Read device model and firmware version of the connected Bodystat device.

Parameters

Parameters	Description
pModel	[out] Receives the device model.
pMajorVersion	[out] Receives the major version of the main firmware in the device.
pMinorVersion	[out] Receives the minor version of the main firmware in the device.
pPsoc2Version	[out] Receives the version of Psoc2 firmware in the device.
pEepromVersion	[out] Receives the version of the EEprom firmware in the device.
pBluetoothInfo	[out] Receives the version of the Bluetooth module in the device.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect previously.

1.2.1.72 BodystatAPI.BodystatAPI.BSReadPrinterAddress@out string

Read the printer address the connected device is paired with (for direct printing). Not all device models support this function (not all support direct printing).

Parameters

Parameters	Description
lpPrinterAddress	[out] String buffer containing the Bluetooth address (BDA) of the printer.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

1.2.1.73 BodystatAPI.BodystatAPI.BSReadPrinterAddress@StringBuilder@int

Read the printer address the connected device is paired with (for direct printing). Not all device models support this function (not all support direct printing).

Parameters

Parameters	Description
lpPrinterAddress	[out] String buffer which receives the printer address.

iPrinterAddressBufferSize	Size of the buffer.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect previously.

1.2.1.74 BodystatAPI.BodystatAPI.BSReadProtocolInfo@out byte@out byte@out byte

Read protocol information from the connected device. Contains version numbers relating to communication protocols in use, data structures and auxiliary information.

Parameters

Parameters	Description
pProtocolVersion	[out] Receives the communication protocol version.
pDataVersion	[out] Receives the data structure version.
pAuxInfo	[out] Receives auxiliary information.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

Generally not required for the majority of SDK users as these details are abstracted by the SDK. You must have connected to the device by calling BSConnect previously.

1.2.1.75 BodystatAPI.BodystatAPI.BSReadSerialNumber@out uint

Read the unique serial number of the connected Bodystat device.

Parameters

Parameters	Description
pSerialNumber	[out] Receives the serial number of the device.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect previously.

1.2.1.76 BodystatAPI.BodystatAPI.BSReadTestBodystat@out int@out int@out int@out int@out int@out float@out float

Perform a quick bio-impedance test measurement on the connected bodystat device. Normal test process is skipped (user is not prompted to enter any subject information). Normal test protocol is disregarded (measurement proceeds regardless of lead status). Test measurement is not stored in device memory.

May give wildly erratic readings if leads are not connected properly. Caller should sanity check results to detect for such scenarios. Results are intentionally not sanity checked to provide a true raw reading and provide caller with total discretion.

Useful for diagnostics or performing measurements instigated by software.

Parameters

Parameters	Description
pZ5	[out] Receives the impedance measured at 5 kHz (result in ohms).
pZ50	[out] Receives the impedance measured at 50 kHz (result in ohms).
pZ100	[out] Receives the impedance measured at 100 kHz (result in ohms).
pZ200	[out] Receives the impedance measured at 200 kHz (result in ohms).
pR50	[out] Receives the resistance measured at 50 kHz (result in ohms).
pX50	[out] Receives the reactance measured at 50 kHz (result in ohms).
pPA50	[out] Receives the phase angle measured at 50 kHz (result in degrees).

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect previously. Caller must check they are working with a Bodystat (rather than Equistat model, Equistat models will respond without error - but with garbage results).

1.2.1.77 BodystatAPI.BodystatAPI.BSResults.BCM

Body cell mass (Quad only)

1.2.1.78 BodystatAPI.BodystatAPI.BSResults.BFMI

Body Fat Mass Index (MDD/Quad only)

1.2.1.79 BodystatAPI.BodystatAPI.BSResults.BMI

Body Mass Index

1.2.1.80 BodystatAPI.BodystatAPI.BSResults.BMR

Basal Metabolic Rate (in kcal)

1.2.1.81 BodystatAPI.BodystatAPI.BSResults.BMRkg

Basal Metabolic Rate per kilogram in (kcal/kg)

1.2.1.82 BodystatAPI.BodystatAPI.BSResults.DryLW

Dry Lean Weight (in kg)

1.2.1.83 BodystatAPI.BodystatAPI.BSResults.ECW

Extra Cellular Water (in litres) (MDD/Quad only)

1.2.1.84 BodystatAPI.BodystatAPI.BSResults.ECWPerc

Extra Cellular Water percentage (MDD/Quad only)

1.2.1.85 BodystatAPI.BodystatAPI.BSResults.EstAvg

Estimated Average Requirement (in kcal)

1.2.1.86 BodystatAPI.BodystatAPI.BSResults.FatKg

Fat (in kg)

1.2.1.87 BodystatAPI.BodystatAPI.BSResults.FatPerc

Fat percentage

1.2.1.88 BodystatAPI.BodystatAPI.BSResults.FFMI

Fat Free Mass Index (MDD/Quad only)

1.2.1.89 BodystatAPI.BodystatAPI.BSResults.ICW

Intra Cellular Water (in litres) (MDD/Quad only)

1.2.1.90 BodystatAPI.BodystatAPI.BSResults.ICWPerc

Intra Cellular Water percentage (MDD/Quad only)

1.2.1.91 BodystatAPI.BodystatAPI.BSResults.Illness

Prediction Marker (TM) - formerly known as Illness Marker (TM) - (Quad only)

1.2.1.92 BodystatAPI.BodystatAPI.BSResults.LeanKg

Lean (in kg)

1.2.1.93 BodystatAPI.BodystatAPI.BSResults.LeanPerc

Lean percentage

1.2.1.94 BodystatAPI.BodystatAPI.BSResults.Nutrition

Nutrition index (Quad only)

1.2.1.95 BodystatAPI.BodystatAPI.BSResults.TBW

Water (in litres) or Total Body Water (in litres)

1.2.1.96 BodystatAPI.BodystatAPI.BSResults.TBWPerc

Water percentage or Total Body Water percentage

1.2.1.97 BodystatAPI.BodystatAPI.BSResults.ThirdSpace

3rd space water (in litres) (Quad only)

1.2.1.98 BodystatAPI.BodystatAPI.BSResults.TotalWeight

Total Weight (in kg)

1.2.1.99 BodystatAPI.BodystatAPI.BSResults.WaistHip

Waist/Hip ratio

1.2.1.100 BodystatAPI.BodystatAPI.BSResults.Wellness

Wellness Marker (TM) (MDD only)

1.2.1.101 BodystatAPI.BodystatAPI.BSWriteCurrentTime

Reset the current date/time of the internal clock on the connected device. Date/time is automatically set to match the current date/time of the PC

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect previously. Locale Daylight Saving Time (DST) rules are observed. Use sister function BSWriteCurrentTime(__time64_t CurrentTime) to set a specific date/time which differs from the PC clock or override DST behaviour.

1.2.1.102 BodystatAPI.BodystatAPI.BSWritePrinterAddress@string@int

Set the address of the printer the connected device should use for direct printing. Programmatic implementation of the inbuilt printer pairing function in the device.

Parameters

Parameters	Description
lpPrinterAddress	String buffer containing the Bluetooth address (BDA) of the printer.
iPrinterAddressBufferSize	Size of the printer address buffer.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

1.2.1.103 BodystatAPI.BodystatAPI.BSWritePrinterAddress@StringBuilder@int

Set the address of the printer the connected device should use for direct printing. Programmatic implementation of the inbuilt printer pairing function in the device.

Parameters

Parameters	Description
lpPrinterAddress	String buffer containing the Bluetooth address (BDA) of the printer.
iBufferSize	Size of the buffer.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

Note not all device models support this function (not all support direct printing).

1.2.1.104 BodystatAPI.BodystatAPI.ESCalculateNormals@ESMeasurement@ESResults@out ESNormals

Calculate normal values and normal ranges for a given animal and their bio-impedance measurement together with the matching results structure for that measurement. You must supply a populated results structure which pertains to the measurement being supplied (obtained from ESCalculateResults).

Parameters

Parameters	Description
pM	The measurement details downloaded from the device (includes model, user inputted parameters and bio-impedance measurement readings).
pR	The results calculated for this measurement previously.
pN	[out] Receives the normal values and normal ranges for this measurement.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

Normals vary by input attributes, measured bio-impedance values and calculated results.

1.2.1.105 BodystatAPI.BodystatAPI.ESCalculateResults@ESMeasurement@out ESResults

Calculate complete result set for a given bio-impedance measurement recorded by the Equistat device.

Parameters

Parameters	Description
pM	The measurement details downloaded from the device (includes model, user inputted parameters and bio-impedance measurement readings).
pR	[out] Receives the complete calculated results for this measurement.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

Results vary by input attributes and measured bio-impedance values.

1.2.1.106 BodystatAPI.BodystatAPI.ESRawData.ESRawData@int

Override the constructor to initialise required variables in the structure. You must ensure you allocate the structure in a manner which calls this function.

Parameters

Parameters	Description
iDontCare	=0

Description

The parameter is not required by the wrapper and is ignored.

1.2.1.107 BodystatAPI.BodystatAPI.ESReadTestEquistat@out int@out int@out int@out int@out int@out int@out int@out float

Perform a quick bio-impedance test measurement on the connected Equistat device. Normal test process is skipped (user is not prompted to enter any subject information). Normal test protocol is disregarded (measurement proceeds regardless of lead status). Test measurement is not stored in device memory.

May give wildly erratic readings if leads are not connected properly. Caller should sanity check results to detect for such scenarios. Results are intentionally not sanity checked to provide a true raw reading and provide caller with total discretion.

Useful for diagnostics or performing measurements instigated by software.

Parameters

Parameters	Description
pZ5	[out] Receives the impedance measured at 5 kHz (result in ohms).
pZ16	[out] Receives the impedance measured at 16 kHz (result in ohms).
pZ24	[out] Receives the impedance measured at 24 kHz (result in ohms).
pZ50	[out] Receives the impedance measured at 50 kHz (result in ohms).
pZ140	[out] Receives the impedance measured at 140 kHz (result in ohms).
pZ200	[out] Receives the impedance measured at 200 kHz (result in ohms).
pZ280	[out] Receives the impedance measured at 280 kHz (result in ohms).
pPA50	[out] Receives the phase angle measured at 50 kHz (result in degress) (ES Pro Only).

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect previously. Caller must check they are working with an Equistat (rather than Bodystat model, Bodystat models will respond without error - but with garbage results).

1.2.1.108 BodystatAPI.BodystatAPI.BSGetBTBodystatDevice@out string@out string@out string@UInt16

Used to get information about any Bodystat device found after a previous call to BSSearchBTDevices.

Parameters

Parameters	Description
lpDeviceName	[out] String buffer to receive the product name of the Bodystat device.
lpBDA	[out] String buffer to receive the Bluetooth hexadecimal address (known as the BDA - unique to each BT module in a device, like a MAC address).
lpComPort	[out] String buffer to receive the device port the device is assigned to.
iTimeout	Timeout multiplier supplied to the Bluetooth radio search API. Controls the amount of time to spend searching for the device. This function will block for this duration so is better called from a thread. Each multiplier increment equates to about 1.28 seconds. Thus a timeout value of 10 = 12.8s. Maximum time permitted by the API is a multiplier of 45 (just under 1 minute), values higher than this will be capped at 45. Default value is 7 (about 10 seconds). = 7

Returns

TRUE if it succeeds, FALSE if it fails.

1.2.1.109 BodystatAPI.BodystatAPI.BSGetBTBodystatDevice@StringBuilder@int@StringBuilder@int@StringBuilder@int@UInt16

Used to get information about any Bodystat device found after a previous call to BSSearchBTDevices.

Parameters

Parameters	Description
lpDeviceName	[out] String buffer to receive the product name of the Bodystat device.
iDeviceNameBufferSize	Size of the device name buffer.
lpBDA	[out] String buffer to receive the Bluetooth hexadecimal address (known as the BDA - unique to each BT module in a device, like a MAC address).
iBDABufferSize	Size of the BDA buffer.
lpComPort	[out] String buffer to receive device/port the unit is assigned to.
iComBufferSize	Size of the com port buffer.
iTimeout	Timeout multiplier supplied to the Bluetooth radio search API. Controls the amount of time to spend searching for the device. This function will block for this duration so is better called from a thread. Each multiplier increment equates to about 1.28 seconds. Thus a timeout value of 10 = 12.8s. Maximum time permitted by the API is a multiplier of 45 (just under 1 minute), values higher than this will be capped at 45. Default value is 7 (about 10 seconds). = 7

Returns

TRUE if it succeeds, FALSE if it fails.

1.2.1.110 BodystatAPI.BodystatAPI.BSSearchBTDevices@out int@out int@bool@IntPtr@UInt16

Search BT radio for any Bodystat devices (paired or otherwise). Existing paired devices will be counted even if they are switched off or out of range. New non-paired units will only be counted if switched on, in range and ready for communication (on main startup screen).

Parameters

Parameters	Description
iNewDevices	[out] The number of new Bodystat devices found (not paired).
iAuthenticatedDevices	[out] The number of authenticated Bodystat devices (paired).
bReportErrors	TRUE to report errors to the user (shown as message boxes, locale constraints apply). FALSE to hide errors from the user (completely silent).
hWndParent	Handle of the parent window.

iTimeout	Timeout multiplier supplied to the Bluetooth radio search API. Controls the amount of time to spend searching for the device. This function will block for this duration so is better called from a thread. Each multiplier increment equates to about 1.28 seconds. Thus a timeout value of 10 = 12.8s. Maximum time permitted by the API is a multiplier of 45 (just under 1 minute), values higher than this will be capped at 45. Default value is 7 (about 10 seconds). = 7

Returns

TRUE if it succeeds, FALSE if it fails.

Remarks

Note this process can be time consuming to complete and executes synchronously. Either call from a threaded function or illustrate operation in progress to the user e.g. by means of an indefinite progress bar or busy mouse cursor, until the task completes or times out.

1.2.1.111 BodystatAPI.BodystatAPI.BSUnAuthenticateBTDevices@bool@IntPtr@UInt16

Unauthenticate (unpair) ALL Bodystat devices from this PC.

Parameters

Parameters	Description
bReportErrors	TRUE to report errors to the user (shown as message boxes, locale constraints apply). FALSE to hide errors from the user.
hWndParent	Handle of the parent window.
iTimeout	Timeout multiplier supplied to the Bluetooth radio search API. Controls the amount of time to spend searching for the device. This function will block for this duration so is better called from a thread. Each multiplier increment equates to about 1.28 seconds. Thus a timeout value of 10 = 12.8s. Maximum time permitted by the API is a multiplier of 45 (just under 1 minute), values higher than this will be capped at 45. Default value is 7 (about 10 seconds). = 7

Returns

TRUE if it succeeds, FALSE if it fails.

Remarks

Note this process can be time consuming to complete and executes synchronously. Either call from a threaded function or illustrate operation in progress to the user e.g. by means of an indefinite progress bar or busy mouse cursor, until the task completes or times out.

1.2.2 Files

The following table lists files in this documentation.

Files

Name	Description
BodystatSDK.cs (↗ see page 80)	Bodystat SDK - C# Wrapper (C) 2010-2014 Bodystat (Isle of Man) Ltd. All rights reserved. Use subject to license. Web: http://www.bodystat.com Email: info@bodystat.com Tel: +44 (0)1624 629 571 Fax: +44 (0)1624 611 544 Version For version information please refer to main BodystatSDK API

Module

C# .NET Wrapper (↗ see page 56)

1.2.2.1 BodystatSDK.cs

Bodystat SDK - C# Wrapper

(C) 2010-2014 Bodystat (Isle of Man) Ltd. All rights reserved.

Use subject to license.

Web: http://www.bodystat.com

Email: info@bodystat.com

Tel: +44 (0)1624 629 571

Fax: +44 (0)1624 611 544

Version

For version information please refer to main BodystatSDK API

Namespaces

Name	Description
BodystatAPI (↗ see page 57)	This is the Bodystat API namespace for our Bodystat DLL wrapper. See main BodystatAPI class members for more information.

1.3 Visual Basic .NET wrapper

Bodystat API Visual Basic .NET wrapper.

Provides a wrapper interface to the Bodystat DLL with VB calls and types you can work with directly in your Visual Basic .NET program. The wrapper handles the necessary marshalling and interop issues required to interface with the native unmanaged Bodystat DLL.

Classes

	Name	Description
	BodystatAPI (see page 81)	This is the main class for our Bodystat DLL wrapper.

Structs, Records, Enums

	Name	Description

	BSAnalysisMode (see page 145)	BSAnalysisMode: Analysis results are available in several formats:
		BSAnalysisModeBC= Body composition analysis
		BSAnalysisModeHN= Hydration analysis
		BSAnalysisModeBoth= Combined body composition and hydration analysis (default)

	BSDeviceFamily (see page 146)	BSDeviceFamily: Enumerated values that group each device / model variant into a given product family. Devices within a given product family share broad capabilities and features. For details regarding capabilities and features refer to the technical documentation for that product family (http://www.bodystat.com)
	BSDeviceModel (see page 147)	BSDeviceModel: Enumerated values that represent the various different models of the Bodystat devices over the years. Suffix: The suffix after the model indicates specific hardware variations regarding connectivity: OPTO indicates model communicates via opto-isolated serial. DIU indicates model communicates via a data interface unit (induction interface) BT indicates model communicates via Bluetooth serial port.
	BSError (see page 149)	BSError: Bodystat Error codes. A list of Bodystat specific error codes.
	BSGender (see page 151)	BSGender: Enumerated values that represent a person's gender.

1.3.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	BodystatAPI (see page 81)	This is the main class for our Bodystat DLL wrapper.

Module

Visual Basic .NET wrapper (see page 80)

1.3.1.1 BodystatAPI Class

This is the main class for our Bodystat DLL wrapper.

Class Hierarchy

BodystatAPI.BodystatAPI

C++

```
class BodystatAPI;
```

C#

```
public class BodystatAPI;
```

Visual Basic

```
Public Class BodystatAPI
```

Java

```
public class BodystatAPI;
```

MATLAB

```
BodystatAPI
```

File

BodystatSDK.vb

BodystatAPI Fields

	Name	Description
◆	BS_RAWDATA_ARRAYSIZE (see page 114)	Number of measurement records BSRawData (see page 96) structure should reserved space for (currently fixed)
◆	ES_RAWDATA_ARRAYSIZE (see page 115)	Number of measurement records ESRawData (see page 109) structure should reserved space for (currently fixed)

BodystatAPI Methods

	Name	Description
◆ S	BSAuthenticateBTDevice (see page 115)	Authenticate (pair) a Bodystat device found during a previous call to BSSearchBTDevices (see page 136). Completely silent implementation.
◆ S	BSAutoSetupBT (see page 116)	Automatic Bluetooth setup. Automatically searches available Bluetooth devices, exchanges passkeys with Bodystat device and enables serial port service required for communication with the device. If multiple Bodystat devices are detected the user is prompted to chose the desired device (model and BT BDA address shown). This function is typically called as a one time setup for the device (e.g. after first installation, or from the hardware configuration page of your application's settings, or when the user obtains a replacement device (upgrades/replacement/etc). Typically, once setup you need only store the device com port returned. You can then subsequently communicate with the... more (see page 116)
◆ S	BSCalculateNormals (see page 117)	Calculate normal values and normal ranges for a given subject and their bio-impedance measurement together with the matching results structure for that measurement. You must supply a populated results structure which pertains to the measurement being supplied (obtained from BSCalculateResults (see page 118)).
◆ S	BSCalculateResults (see page 118)	Calculate complete result set for a given bio-impedance measurement recorded by the Bodystat device.
◆ S	BSCloseComport (see page 119)	Close com port.
◆ S	BSConnect (see page 120)	Connect to the Bodystat device and ensure the device is responding.
◆ S	BSGetBTBodystatDevice (see page 120)	Used to get information about any Bodystat device found after a previous call to BSSearchBTDevices (see page 136).
◆ S	BSGetBTStackInfo (see page 121)	Retrieve information about the Bluetooth stack. Useful for diagnostics.
◆ S	BSGetDeviceFamily (see page 122)	Determine the device family of a specific model.
◆ S	BSGetDeviceModelName (see page 123)	Determine the full product name of a specific model.
◆ S	BSGetDeviceModelNameShort (see page 124)	Determine the short product name of a specific model.
◆ S	BSGetSdkLibraryVersion (see page 124)	Determine the Bodystat SDK library (DLL) version.
◆ S	BSIsBTAvailable (see page 125)	Check if a supported BT radio is available on this computer. Checks for presence of USB Bluetooth dongles and inbuilt Bluetooth modules in the PC. No communication with Bodystat devices takes place during this query.

	BSOpenComport (see page 126)	Open com port to the device ready for communication.
	BSReadCalibrationTime (see page 127)	Read the calibration date from the connected device (when the unit was last calibrated). Only the date element of this value should be shown. Exact time value is not relevant.
	BSReadCurrentTime (see page 128)	Read the current time from the internal clock of the connected device. Note not all device models support this function (not all have an internal clock).
	BSReadModelVersion (see page 130)	Read device model and firmware version of the connected Bodystat device.
	BSReadPrinterAddress (see page 131)	Read the printer address the connected device is paired with (for direct printing). Not all device models support this function (not all support direct printing).
	BSReadProtocolInfo (see page 132)	Read protocol information from the connected device. Contains version numbers relating to communication protocols in use, data structures and auxiliary information.
	BSReadSerialNumber (see page 133)	Read the unique serial number of the connected Bodystat device.
	BSReadStoredTestData (see page 133)	Download test data from the connected device. Retrieves the last 100 test measurements (records) stored in the memory of the device. Downloading data does not affect the results stored in the memory - they remain unchanged. Note: you should store downloaded data locally if it is required permanently. Most devices store only the last 100 test records before overwriting the oldest records automatically.
	BSReadTestBodystat (see page 134)	Perform a quick bio-impedance test measurement on the connected bodystat device. Normal test process is skipped (user is not prompted to enter any subject information). Normal test protocol is disregarded (measurement proceeds regardless of lead status). Test measurement is not stored in device memory. May give wildly erratic readings if leads are not connected properly. Caller should sanity check results to detect for such scenarios. Results are intentionally not sanity checked to provide a true raw reading and provide caller with total discretion. Useful for diagnostics or performing measurements instigated by software.
	BSSearchBTDevices (see page 136)	Search BT radio for any Bodystat devices (paired or otherwise). Existing paired devices will be counted even if they are switched off or out of range. New non-paired units will only be counted if switched on, in range and ready for communication (on main startup screen).
	BSUnAuthenticateBTDevices (see page 137)	Unauthenticate (unpair) ALL Bodystat devices from this PC.
	BSWriteCurrentTime (see page 138)	Reset the current date/time of the internal clock on the connected device. Date/time is automatically set to match the current date/time of the PC
	BSWriteCurrentTimeAs (see page 138)	Reset the current date/time of the internal clock on the connected device. Date/time is set to a specific value of your choosing.
	BSWritePrinterAddress (see page 140)	Set the address of the printer the connected device should use for direct printing. Programmatic implementation of the inbuilt printer pairing function in the device.
	ESCalculateNormals (see page 141)	Calculate normal values and normal ranges for a given animal and their bio-impedance measurement together with the matching results structure for that measurement. You must supply a populated results structure which pertains to the measurement being supplied (obtained from ESCalculateResults (see page 142)).
	ESCalculateResults (see page 142)	Calculate complete result set for a given bio-impedance measurement recorded by the Equistat device.

	ESReadStoredTestData (see page 142)	Download test data from the connected device. Retrieves the last 100 test measurements (records) stored in the memory of the device. Downloading data does not affect the results stored in the memory - they remain unchanged. Note: you should store downloaded data locally if it is required permanently. Most devices store only the last 100 test records before overwriting the oldest records automatically.
	ESReadTestEquistat (see page 143)	Perform a quick bio-impedance test measurement on the connected Equistat device. Normal test process is skipped (user is not prompted to enter any subject information). Normal test protocol is disregarded (measurement proceeds regardless of lead status). Test measurement is not stored in device memory. May give wildly erratic readings if leads are not connected properly. Caller should sanity check results to detect for such scenarios. Results are intentionally not sanity checked to provide a true raw reading and provide caller with total discretion. Useful for diagnostics or performing measurements instigated by software.

BodystatAPI Structures

	Name	Description
	BSMeasurement (see page 84)	BSMeasurement: Measurement structure. Each test measurement recorded in the device is stored one of these structures. Holds the user inputted information about the subject, together with bio-impedance readings measured by the device (raw electrical readings only, not full results).
	BSNormals (see page 93)	BSNormals: Bodystat Normals structure. Holds the normal values or normal ranges for any given test measurement.
	BSRawData (see page 96)	BSRawData: Bodystat Raw Data structure. Used when downloading test measurements from the device. Contains a fixed array of measurement records together with information about the number of records occupied within the array and test record (see page 98) starting number.
	BSResults (see page 99)	BSResults: Bodystat Results structure. Holds the calculated results for any given test measurement.
	ESMeasurement (see page 101)	ESMeasurement: Measurement structure. Each test measurement recorded in the device is stored one of these structures. Holds the user inputted information about the subject, together with bio-impedance readings measured by the device (raw electrical readings only, not full results).
	ESNormals (see page 108)	ESNormals: Equistat Normals structure. Holds the normal values or normal ranges for any given test measurement.
	ESRawData (see page 109)	ESRawData: Equistat Raw Data structure. Used when downloading test measurements from the device. Contains a fixed array of measurement records together with information about the number of records occupied within the array and test record (see page 112) starting number.
	ESResults (see page 112)	ESResults: Equistst Results structure. Holds the calculated results for any given test measurement.
	ImpData (see page 114)	Represents one frequency measurement

1.3.1.1.1 BodystatAPI Structures

1.3.1.1.1.1 BodystatAPI.BSMeasurement Structure

BSMeasurement: Measurement structure. Each test measurement recorded in the device is stored one of these structures. Holds the user inputted information about the subject, together with bio-impedance readings measured by the device (raw electrical readings only, not full results).

C++

```
[System.Runtime.InteropServices.StructLayoutAttribute(System.Runtime.InteropServices.LayoutK
ind.Sequential)]
struct BSMeasurement {
  BSDeviceModel iDeviceModel;
  UInteger ulDeviceSerialNumber;
  Int64 llTestDate;
  BSGender iGender;
  Integer iAge;
  Integer iHeight;
  Single fWeight;
  Integer iActivity;
  Integer iWaist;
  Integer iHip;
  Integer iZ_5kHz;
  Integer iZ_50kHz;
  Integer iZ_100kHz;
  Integer iZ_200kHz;
  Integer iR_50kHz;
  Single fX_50kHz;
  Single fPA_50kHz;
  Integer iFrequencies;
  ImpData pMultifreqData;
};
```

C#

```
[System.Runtime.InteropServices.StructLayoutAttribute(System.Runtime.InteropServices.LayoutK
ind.Sequential)]
public struct BSMeasurement {
  public BSDeviceModel iDeviceModel;
  public UInteger ulDeviceSerialNumber;
  public Int64 llTestDate;
  public BSGender iGender;
  public Integer iAge;
  public Integer iHeight;
  public Single fWeight;
  public Integer iActivity;
  public Integer iWaist;
  public Integer iHip;
  public Integer iZ_5kHz;
  public Integer iZ_50kHz;
  public Integer iZ_100kHz;
  public Integer iZ_200kHz;
  public Integer iR_50kHz;
  public Single fX_50kHz;
  public Single fPA_50kHz;
  public Integer iFrequencies;
  public ImpData() pMultifreqData;
}
```

Visual Basic

```
<System.Runtime.InteropServices.StructLayoutAttribute(System.Runtime.InteropServices.LayoutK
ind.Sequential)>
Public Structure BSMeasurement
  Public iDeviceModel As BSDeviceModel
  Public ulDeviceSerialNumber As UInteger
  Public llTestDate As Int64
  Public iGender As BSGender
  Public iAge As Integer
  Public iHeight As Integer
  Public fWeight As Single
  Public iActivity As Integer
  Public iWaist As Integer
  Public iHip As Integer
  Public iZ_5kHz As Integer
  Public iZ_50kHz As Integer
  Public iZ_100kHz As Integer
  Public iZ_200kHz As Integer
  Public iR_50kHz As Integer
  Public fX_50kHz As Single
```

```
    Public fPA_50kHz As Single
    Public iFrequencies As Integer
    Public pMultifreqData As ImpData()
  End Structure
```

Java

```
[System.Runtime.InteropServices.StructLayoutAttribute(System.Runtime.InteropServices.LayoutK
ind.Sequential)]
public class BSMeasurement;
```

MATLAB

```
BSMeasurement
```

File

BodystatSDK.vb

BSMeasurement Fields

	Name	Description
◆	fPA_50kHz (see page 86)	Phase angle measured at 50 kHz (result in degrees) (MDD/Quad only)
◆	fWeight (see page 87)	Inputted weight of the subject (kilograms)
◆	fX_50kHz (see page 87)	Reactance measured at 50 kHz (result in ohms) (MDD/Quad only)
◆	iActivity (see page 87)	Inputted physical activity level of the subject (range 1 to 5: 1=very low 2=low/medium 3=medium 4=medium/high 5=very high)
◆	iAge (see page 88)	Inputted age of the subject (years)
◆	iDeviceModel (see page 88)	Device model which performed the measurement
◆	iFrequencies (see page 88)	Number of frequencies used 11/50
◆	iGender (see page 89)	Inputted gender of the subject
◆	iHeight (see page 89)	Inputted height of the subject (centimeters)
◆	iHip (see page 89)	Inputted hip size of the subject (centimeters)
◆	iR_50kHz (see page 90)	Resistance measured at 50 kHz (result in ohms) (MDD/Quad only)
◆	iWaist (see page 90)	Inputted waist size of the subject (centimeters)
◆	iZ_100kHz (see page 90)	Impedance measured at 100 kHz (result in ohms) (Quad only)
◆	iZ_200kHz (see page 91)	Impedance measured at 200 kHz (result in ohms) (Quad only)
◆	iZ_50kHz (see page 91)	Impedance measured at 50 kHz (result in ohms)
◆	iZ_5kHz (see page 91)	Impedance measured at 5 kHz (result in ohms) (MDD/Quad only)
◆	llTestDate (see page 92)	Date & time the test measurement was performed (Windows 64-bit time structure) (MDD/Quad only)
◆	pMultifreqData (see page 92)	Pointer to ImpData (see page 114) array
◆	ulDeviceSerialNumber (see page 92)	Unique serial number of the device which performed the measurement

BSMeasurement Properties

	Name	Description
▨	tTestDate (see page 93)	Convert underlying 64-bit time structure to/from Visual Basic Date/Time

1.3.1.1.1.1.1 BSMeasurement Fields

1.3.1.1.1.1.1.1 BodystatAPI.BSMeasurement.fPA_50kHz Field

C++

```
Single fPA_50kHz;
```

C#

```
public Single fPA_50kHz;
```

Visual Basic

```
Public fPA_50kHz As Single
```

Java

```
public Single fPA_50kHz;
```

MATLAB

```
fPA_50kHz
```

Description

Phase angle measured at 50 kHz (result in degrees) (MDD/Quad only)

1.3.1.1.1.1.1.2 BodystatAPI.BSMeasurement.fWeight Field

C++

```
Single fWeight;
```

C#

```
public Single fWeight;
```

Visual Basic

```
Public fWeight As Single
```

Java

```
public Single fWeight;
```

MATLAB

```
fWeight
```

Description

Inputted weight of the subject (kilograms)

1.3.1.1.1.1.1.3 BodystatAPI.BSMeasurement.fX_50kHz Field

C++

```
Single fX_50kHz;
```

C#

```
public Single fX_50kHz;
```

Visual Basic

```
Public fX_50kHz As Single
```

Java

```
public Single fX_50kHz;
```

MATLAB

```
fX_50kHz
```

Description

Reactance measured at 50 kHz (result in ohms) (MDD/Quad only)

1.3.1.1.1.1.1.4 BodystatAPI.BSMeasurement.iActivity Field

C++

```
Integer iActivity;
```

C#

```
public Integer iActivity;
```

Visual Basic

```
Public iActivity As Integer
```

Java

```
public Integer iActivity;
```

MATLAB

```
iActivity
```

Description

Inputted physical activity level of the subject (range 1 to 5: 1=very low 2=low/medium 3=medium 4=medium/high 5=very high)

1.3.1.1.1.1.1.5 BodystatAPI.BSMeasurement.iAge Field

C++

```
Integer iAge;
```

C#

```
public Integer iAge;
```

Visual Basic

```
Public iAge As Integer
```

Java

```
public Integer iAge;
```

MATLAB

```
iAge
```

Description

Inputted age of the subject (years)

1.3.1.1.1.1.1.6 BodystatAPI.BSMeasurement.iDeviceModel Field

C++

```
BSDeviceModel iDeviceModel;
```

C#

```
public BSDeviceModel iDeviceModel;
```

Visual Basic

```
Public iDeviceModel As BSDeviceModel
```

Java

```
public BSDeviceModel iDeviceModel;
```

MATLAB

```
iDeviceModel
```

Description

Device model which performed the measurement

1.3.1.1.1.1.1.7 BodystatAPI.BSMeasurement.iFrequencies Field

C++

```
Integer iFrequencies;
```

C#

```
public Integer iFrequencies;
```

Visual Basic

```
Public iFrequencies As Integer
```

Java

```
public Integer iFrequencies;
```

MATLAB

```
iFrequencies
```

Description

Number of frequencies used 11/50

1.3.1.1.1.1.1.8 BodystatAPI.BSMeasurement.iGender Field

C++

```
BSGender iGender;
```

C#

```
public BSGender iGender;
```

Visual Basic

```
Public iGender As BSGender
```

Java

```
public BSGender iGender;
```

MATLAB

```
iGender
```

Description

Inputted gender of the subject

1.3.1.1.1.1.1.9 BodystatAPI.BSMeasurement.iHeight Field

C++

```
Integer iHeight;
```

C#

```
public Integer iHeight;
```

Visual Basic

```
Public iHeight As Integer
```

Java

```
public Integer iHeight;
```

MATLAB

```
iHeight
```

Description

Inputted height of the subject (centimeters)

1.3.1.1.1.1.1.10 BodystatAPI.BSMeasurement.iHip Field

C++

```
Integer iHip;
```

C#

```
public Integer iHip;
```

Visual Basic

```
Public iHip As Integer
```

Java

```
public Integer iHip;
```

MATLAB

```
iHip
```

Description

Inputted hip size of the subject (centimeters)

1.3.1.1.1.1.1.11 BodystatAPI.BSMeasurement.iR_50kHz Field

C++

```
Integer iR_50kHz;
```

C#

```
public Integer iR_50kHz;
```

Visual Basic

```
Public iR_50kHz As Integer
```

Java

```
public Integer iR_50kHz;
```

MATLAB

```
iR_50kHz
```

Description

Resistance measured at 50 kHz (result in ohms) (MDD/Quad only)

1.3.1.1.1.1.1.12 BodystatAPI.BSMeasurement.iWaist Field

C++

```
Integer iWaist;
```

C#

```
public Integer iWaist;
```

Visual Basic

```
Public iWaist As Integer
```

Java

```
public Integer iWaist;
```

MATLAB

```
iWaist
```

Description

Inputted waist size of the subject (centimeters)

1.3.1.1.1.1.1.13 BodystatAPI.BSMeasurement.iZ_100kHz Field

C++

```
Integer iZ_100kHz;
```

C#

```
public Integer iZ_100kHz;
```

Visual Basic

```
Public iZ_100kHz As Integer
```

Java

```
public Integer iZ_100kHz;
```

MATLAB

```
iZ_100kHz
```

Description

Impedance measured at 100 kHz (result in ohms) (Quad only)

1.3.1.1.1.1.1.14 BodystatAPI.BSMeasurement.iZ_200kHz Field

C++

```
Integer iZ_200kHz;
```

C#

```
public Integer iZ_200kHz;
```

Visual Basic

```
Public iZ_200kHz As Integer
```

Java

```
public Integer iZ_200kHz;
```

MATLAB

```
iZ_200kHz
```

Description

Impedance measured at 200 kHz (result in ohms) (Quad only)

1.3.1.1.1.1.1.15 BodystatAPI.BSMeasurement.iZ_50kHz Field

C++

```
Integer iZ_50kHz;
```

C#

```
public Integer iZ_50kHz;
```

Visual Basic

```
Public iZ_50kHz As Integer
```

Java

```
public Integer iZ_50kHz;
```

MATLAB

```
iZ_50kHz
```

Description

Impedance measured at 50 kHz (result in ohms)

1.3.1.1.1.1.1.16 BodystatAPI.BSMeasurement.iZ_5kHz Field

C++

```
Integer iZ_5kHz;
```

C#

```
public Integer iZ_5kHz;
```

Visual Basic

```
Public iZ_5kHz As Integer
```

Java

```
public Integer iZ_5kHz;
```

MATLAB

```
iZ_5kHz
```

Description

Impedance measured at 5 kHz (result in ohms) (MDD/Quad only)

1.3.1.1.1.1.1.17 BodystatAPI.BSMeasurement.llTestDate Field

C++

```
Int64 llTestDate;
```

C#

```
public Int64 llTestDate;
```

Visual Basic

```
Public llTestDate As Int64
```

Java

```
public Int64 llTestDate;
```

MATLAB

```
llTestDate
```

Description

Date & time the test measurement was performed (Windows 64-bit time structure) (MDD/Quad only)

1.3.1.1.1.1.1.18 BodystatAPI.BSMeasurement.pMultifreqData Field

C++

```
ImpData pMultifreqData;
```

C#

```
public ImpData() pMultifreqData;
```

Visual Basic

```
Public pMultifreqData As ImpData()
```

Java

```
public ImpData() pMultifreqData;
```

MATLAB

```
pMultifreqData
```

Description

Pointer to ImpData (see page 114) array

1.3.1.1.1.1.1.19 BodystatAPI.BSMeasurement.ulDeviceSerialNumber Field

C++

```
UInteger ulDeviceSerialNumber;
```

C#

```
public UInteger ulDeviceSerialNumber;
```

Visual Basic

```
Public ulDeviceSerialNumber As UInteger
```

Java

```
public UInteger ulDeviceSerialNumber;
```

MATLAB

```
ulDeviceSerialNumber
```

Description

Unique serial number of the device which performed the measurement

1.3.1.1.1.1.2 BSMeasurement Properties

1.3.1.1.1.1.2.1 BodystatAPI.BSMeasurement.tTestDate Property

C++

```
__property DateTime tTestDate;
```

C#

```
public DateTime tTestDate;
```

Visual Basic

```
Public Property tTestDate() As DateTime
```

Java

```
tTestDate
```

MATLAB

```
tTestDate
```

Description

Convert underlying 64-bit time structure to/from Visual Basic Date/Time

1.3.1.1.1.2 BodystatAPI.BodystatAPI.BSNormals Structure

BSNormals: Bodystat Normals structure. Holds the normal values or normal ranges for any given test measurement.

C++

```
[StructLayout(LayoutKind.Sequential)]
struct BSNormals {
  Integer iFatPerc_L;
  Integer iFatPerc_H;
  Integer iFatKg_L;
  Integer iFatKg_H;
  Integer iLeanPerc_L;
  Integer iLeanPerc_H;
  Integer iLeanKg_L;
  Integer iLeanKg_H;
  Integer iTotalWeight_L;
  Integer iTotalWeight_H;
  Integer iTotalWeightMethod;
  Integer iTBWPerc_L;
  Integer iTBWPerc_H;
  Integer iTBW_L;
  Integer iTBW_H;
  Integer iECWPerc_Norm;
  Integer iICWPerc_Norm;
  Integer iBFMI_L;
  Integer iBFMI_H;
  Integer iFFMI_L;
  Integer iFFMI_H;
  Single fNutrition_Norm;
  Single fIllness_L;
  Single fIllness_H;
  Integer iBMI_L;
  Integer iBMI_H;
```

1

```
    Single fWaistHip_Norm;
    Single fWellness_L;
    Single fWellness_H;
  };
```

C#

```
  [StructLayout(LayoutKind.Sequential)]
  public struct BSNormals {
    public Integer iFatPerc_L;
    public Integer iFatPerc_H;
    public Integer iFatKg_L;
    public Integer iFatKg_H;
    public Integer iLeanPerc_L;
    public Integer iLeanPerc_H;
    public Integer iLeanKg_L;
    public Integer iLeanKg_H;
    public Integer iTotalWeight_L;
    public Integer iTotalWeight_H;
    public Integer iTotalWeightMethod;
    public Integer iTBWPerc_L;
    public Integer iTBWPerc_H;
    public Integer iTBW_L;
    public Integer iTBW_H;
    public Integer iECWPerc_Norm;
    public Integer iICWPerc_Norm;
    public Integer iBFMI_L;
    public Integer iBFMI_H;
    public Integer iFFMI_L;
    public Integer iFFMI_H;
    public Single fNutrition_Norm;
    public Single fIllness_L;
    public Single fIllness_H;
    public Integer iBMI_L;
    public Integer iBMI_H;
    public Single fWaistHip_Norm;
    public Single fWellness_L;
    public Single fWellness_H;
  }
```

Visual Basic

```
  <StructLayout(LayoutKind.Sequential)>
  Public Structure BSNormals
    Public iFatPerc_L As Integer
    Public iFatPerc_H As Integer
    Public iFatKg_L As Integer
    Public iFatKg_H As Integer
    Public iLeanPerc_L As Integer
    Public iLeanPerc_H As Integer
    Public iLeanKg_L As Integer
    Public iLeanKg_H As Integer
    Public iTotalWeight_L As Integer
    Public iTotalWeight_H As Integer
    Public iTotalWeightMethod As Integer
    Public iTBWPerc_L As Integer
    Public iTBWPerc_H As Integer
    Public iTBW_L As Integer
    Public iTBW_H As Integer
    Public iECWPerc_Norm As Integer
    Public iICWPerc_Norm As Integer
    Public iBFMI_L As Integer
    Public iBFMI_H As Integer
    Public iFFMI_L As Integer
    Public iFFMI_H As Integer
    Public fNutrition_Norm As Single
    Public fIllness_L As Single
    Public fIllness_H As Single
    Public iBMI_L As Integer
    Public iBMI_H As Integer
    Public fWaistHip_Norm As Single
    Public fWellness_L As Single
    Public fWellness_H As Single
```

```
End Structure
```

Java

```
[StructLayout(LayoutKind.Sequential)]
public class BSNormals;
```

MATLAB

```
BSNormals
```

File

BodystatSDK.vb

Members

Members	Description
public Integer iFatPerc_L;	Normal range for fat percentage - (low end)
public Integer iFatPerc_H;	Normal range for fat percentage - (high end)
public Integer iFatKg_L;	Normal range for fat (in kg) - (low end)
public Integer iFatKg_H;	Normal range for fat (in kg) - (high end)
public Integer iLeanPerc_L;	Normal range for lean percentage - (low end)
public Integer iLeanPerc_H;	Normal range for lean percentage - (high end)
public Integer iLeanKg_L;	Normal range for lean (in kg) - (low end)
public Integer iLeanKg_H;	Normal range for lean (in kg) - (high end)
public Integer iTotalWeight_L;	Normal range for Total Weight (in kg) - (low end)
public Integer iTotalWeight_H;	Normal range for Total Weight (in kg) - (high end)
public Integer iTotalWeightMethod;	Method used when calculating Total Weight normal range (0 = Composition Method, 1 = BMI Method)
public Integer iTBWPerc_L;	Normal range for Total Body Water percentage - (low end)
public Integer iTBWPerc_H;	Normal range for Total Body Water percentage - (high end)
public Integer iTBW_L;	Normal range for Total Body Water (in litres) - (low end)
public Integer iTBW_H;	Normal range for Total Body Water (in litres) - (high end)
public Integer iECWPerc_Norm;	Normal value for ECW Percentage (MDD/Quad only)
public Integer iICWPerc_Norm;	Normal value for ICW Percentage (MDD/Quad only)
public Integer iBFMI_L;	Normal range for Body Fat Mass Index (MDD/Quad only) - (low end)
public Integer iBFMI_H;	Normal range for Body Fat Mass Index (MDD/Quad only) - (high end)
public Integer iFFMI_L;	Normal range for Fat Free Mass Index (MDD/Quad only) - (low end)
public Integer iFFMI_H;	Normal range for Fat Free Mass Index (MDD/Quad only) - (high end)
public Single fNutrition_Norm;	Normal value for Nutrition Index (Quad only)
public Single fIllness_L;	Normal range for Prediction Marker (TM) - formerly known as Illness Marker (TM) - (Quad only) - (low end)
public Single fIllness_H;	Normal range for Prediction Marker (TM) - formerly known as Illness Marker (TM) - (Quad only) - (high end)
public Integer iBMI_L;	Normal range for Body Mass Index - (low end)
public Integer iBMI_H;	Normal range for Body Mass Index - (high end)
public Single fWaistHip_Norm;	Normal value for Waist/Hip ratio
public Single fWellness_L;	Normal range for Wellness Marker (TM) (MDD only) - (low end)
public Single fWellness_H;	Normal range for Wellness Marker (TM) (MDD only) - (high end)

1.3.1.1.1.3 BodystatAPI.BSRawData Structure

BSRawData: Bodystat Raw Data structure. Used when downloading test measurements from the device. Contains a fixed array of measurement records together with information about the number of records occupied within the array and test record (see page 98) starting number.

C++

```
[StructLayout(LayoutKind.Sequential)]
struct BSRawData {
  Integer iRecordArraySize;
  [System.Runtime.InteropServices.MarshalAsAttribute(System.Runtime.InteropServices.Unmanage
dType.ByValArray,
SizeConst:=BS_RAWDATA_ARRAYSIZE,
ArraySubType:=System.Runtime.InteropServices.UnmanagedType.Struct)]
BSMeasurement record;
  Integer iTotalNumRecs;
  Integer ulFirstTestNum;
};
```

C#

```
[StructLayout(LayoutKind.Sequential)]
public struct BSRawData {
  public Integer iRecordArraySize;
  [System.Runtime.InteropServices.MarshalAsAttribute(System.Runtime.InteropServices.Unmanage
dType.ByValArray,
SizeConst:=BS_RAWDATA_ARRAYSIZE,
ArraySubType:=System.Runtime.InteropServices.UnmanagedType.Struct)]
public BSMeasurement() record;
  public Integer iTotalNumRecs;
  public Integer ulFirstTestNum;
}
```

Visual Basic

```
<StructLayout(LayoutKind.Sequential)>
Public Structure BSRawData
  Public iRecordArraySize As Integer
  <System.Runtime.InteropServices.MarshalAsAttribute(System.Runtime.InteropServices.Unmanage
dType.ByValArray,
SizeConst:=BS_RAWDATA_ARRAYSIZE,
ArraySubType:=System.Runtime.InteropServices.UnmanagedType.Struct)>
Public record As BSMeasurement()
  Public iTotalNumRecs As Integer
  Public ulFirstTestNum As Integer
End Structure
```

Java

```
[StructLayout(LayoutKind.Sequential)]
public class BSRawData;
```

MATLAB

```
BSRawData
```

File

BodystatSDK.vb

Remarks

Most models index 1000 records and store the last 100 at any given time.

Methods

	Name	Description
	New (see page 97)	Override the constructor to initialise required variables in the structure. You must ensure you allocate the structure in a manner which calls this function.

BSRawData Fields

	Name	Description
◆	iRecordArraySize (see page 97)	Maximum number of records this structure has reserved space for (currently fixed)
◆	iTotalNumRecs (see page 98)	Number of populated records in the array
◆	record (see page 98)	Array of test measurement data
◆	ulFirstTestNum (see page 99)	Test number of the first record (see page 98) stored in the array.

1.3.1.1.1.3.1 BodystatAPI.BSRawData.New Constructor

Override the constructor to initialise required variables in the structure. You must ensure you allocate the structure in a manner which calls this function.

C++

```
BSRawData(ByVal Integer dontcare);
```

C#

```
public New(ByVal Integer dontcare);
```

Visual Basic

```
Public Sub New(ByVal dontcare As Integer)
```

Java

```
public BSRawData(ByVal Integer dontcare);
```

MATLAB

```
New
```

Parameters

Parameters	Description
ByVal dontcare As Integer	Ignored: VB requires New() constructors to specify a non-optional parameter for structures. The parameter is not required by the wrapper and is ignored.

Body Source

```
Public Sub New(ByVal dontcare As Integer)
    Me.iRecordArraySize = BS_RAWDATA_ARRAYSIZE  ' Value must be populated with size of
array (currently fixed)
    Me.ulFirstTestNum = 0
    Me.iTotalNumRecs = 0
End Sub
```

1.3.1.1.1.3.2 BSRawData Fields

1.3.1.1.1.3.2.1 BodystatAPI.BSRawData.iRecordArraySize Field

C++

```
Integer iRecordArraySize;
```

C#

```
public Integer iRecordArraySize;
```

Visual Basic

```
Public iRecordArraySize As Integer
```

Java

```
public Integer iRecordArraySize;
```

MATLAB

```
iRecordArraySize
```

Description

Maximum number of records this structure has reserved space for (currently fixed)

1.3.1.1.1.3.2.2 BodystatAPI.BSRawData.iTotalNumRecs Field

C++

```
Integer iTotalNumRecs;
```

C#

```
public Integer iTotalNumRecs;
```

Visual Basic

```
Public iTotalNumRecs As Integer
```

Java

```
public Integer iTotalNumRecs;
```

MATLAB

```
iTotalNumRecs
```

Description

Number of populated records in the array

1.3.1.1.1.3.2.3 BodystatAPI.BSRawData.record Field

C++

```
[System.Runtime.InteropServices.MarshalAsAttribute(System.Runtime.InteropServices.UnmanagedT
ype.ByValArray,
SizeConst:=BS_RAWDATA_ARRAYSIZE,
ArraySubType:=System.Runtime.InteropServices.UnmanagedType.Struct)]
BSMeasurement record;
```

C#

```
[System.Runtime.InteropServices.MarshalAsAttribute(System.Runtime.InteropServices.UnmanagedT
ype.ByValArray,
SizeConst:=BS_RAWDATA_ARRAYSIZE,
ArraySubType:=System.Runtime.InteropServices.UnmanagedType.Struct)]
public BSMeasurement() record;
```

Visual Basic

```
<System.Runtime.InteropServices.MarshalAsAttribute(System.Runtime.InteropServices.UnmanagedT
ype.ByValArray,
SizeConst:=BS_RAWDATA_ARRAYSIZE,
ArraySubType:=System.Runtime.InteropServices.UnmanagedType.Struct)>
Public record As BSMeasurement()
```

Java

```
[System.Runtime.InteropServices.MarshalAsAttribute(System.Runtime.InteropServices.UnmanagedT
ype.ByValArray,
SizeConst:=BS_RAWDATA_ARRAYSIZE,
ArraySubType:=System.Runtime.InteropServices.UnmanagedType.Struct)]
public BSMeasurement() record;
```

MATLAB

```
record
```

Description

Array of test measurement data

1.3.1.1.1.3.2.4 BodystatAPI.BSRawData.ulFirstTestNum Field

C++

```
Integer ulFirstTestNum;
```

C#

```
public Integer ulFirstTestNum;
```

Visual Basic

```
Public ulFirstTestNum As Integer
```

Java

```
public Integer ulFirstTestNum;
```

MATLAB

```
ulFirstTestNum
```

Description

Test number of the first record (see page 98) stored in the array.

1.3.1.1.1.4 BodystatAPI.BodystatAPI.BSResults Structure

BSResults: Bodystat Results structure. Holds the calculated results for any given test measurement.

C++

```
[StructLayout(LayoutKind.Sequential)]
struct BSResults {
  Single fFatPerc;
  Single fFatKg;
  Single fLeanPerc;
  Single fLeanKg;
  Single fTotalWeight;
  Single fDryLW;
  Single fTBWPerc;
  Single fTBW;
  Single fECWPerc;
  Single fECW;
  Single fICWPerc;
  Single fICW;
  Single fBCM;
  Single fThirdSpace;
  Single fNutrition;
  Single fIllness;
  Single fBMR;
  Single fBMRkg;
  Single fEstAvg;
  Single fBMI;
  Single fBFMI;
  Single fFFMI;
  Single fWaistHip;
  Single fWellness;
  Single fECW_Legacy;
  Single fTBW_Legacy;
  Single fOHY;
  Single fSkMuscle;
  Single fCm;
  Single fRext;
  Single fRint;
  Single fFc;
  Single fAlpha;
};
```

C#

```
[StructLayout(LayoutKind.Sequential)]
public struct BSResults {
  public Single fFatPerc;
```

```
    public Single fFatKg;
    public Single fLeanPerc;
    public Single fLeanKg;
    public Single fTotalWeight;
    public Single fDryLW;
    public Single fTBWPerc;
    public Single fTBW;
    public Single fECWPerc;
    public Single fECW;
    public Single fICWPerc;
    public Single fICW;
    public Single fBCM;
    public Single fThirdSpace;
    public Single fNutrition;
    public Single fIllness;
    public Single fBMR;
    public Single fBMRkg;
    public Single fEstAvg;
    public Single fBMI;
    public Single fBFMI;
    public Single fFFMI;
    public Single fWaistHip;
    public Single fWellness;
    public Single fECW_Legacy;
    public Single fTBW_Legacy;
    public Single fOHY;
    public Single fSkMuscle;
    public Single fCm;
    public Single fRext;
    public Single fRint;
    public Single fFc;
    public Single fAlpha;
  }
```

Visual Basic

```
  <StructLayout(LayoutKind.Sequential)>
  Public Structure BSResults
    Public fFatPerc As Single
    Public fFatKg As Single
    Public fLeanPerc As Single
    Public fLeanKg As Single
    Public fTotalWeight As Single
    Public fDryLW As Single
    Public fTBWPerc As Single
    Public fTBW As Single
    Public fECWPerc As Single
    Public fECW As Single
    Public fICWPerc As Single
    Public fICW As Single
    Public fBCM As Single
    Public fThirdSpace As Single
    Public fNutrition As Single
    Public fIllness As Single
    Public fBMR As Single
    Public fBMRkg As Single
    Public fEstAvg As Single
    Public fBMI As Single
    Public fBFMI As Single
    Public fFFMI As Single
    Public fWaistHip As Single
    Public fWellness As Single
    Public fECW_Legacy As Single
    Public fTBW_Legacy As Single
    Public fOHY As Single
    Public fSkMuscle As Single
    Public fCm As Single
    Public fRext As Single
    Public fRint As Single
    Public fFc As Single
    Public fAlpha As Single
  End Structure
```

Java

```
[StructLayout(LayoutKind.Sequential)]
public class BSResults;
```

MATLAB

```
BSResults
```

File

BodystatSDK.vb

Members

Members	Description
public Single fFatPerc;	Fat percentage
public Single fFatKg;	Fat (in kg)
public Single fLeanPerc;	Lean percentage
public Single fLeanKg;	Lean (in kg)
public Single fTotalWeight;	Total Weight (in kg)
public Single fDryLW;	Dry Lean Weight (in kg)
public Single fTBWPerc;	Water percentage or Total Body Water percentage
public Single fTBW;	Water (in litres) or Total Body Water (in litres)
public Single fECWPerc;	Extra Cellular Water percentage (MDD/Quad only)
public Single fECW;	Extra Cellular Water (in litres) (MDD/Quad only)
public Single fICWPerc;	Intra Cellular Water percentage (MDD/Quad only)
public Single fICW;	Intra Cellular Water (in litres) (MDD/Quad only)
public Single fBCM;	Body cell mass (Quad only)
public Single fThirdSpace;	3rd space water (in litres) (Quad only)
public Single fNutrition;	Nutrition index (Quad only)
public Single fIllness;	Prediction Marker (TM) - formerly known as Illness Marker (TM) (Quad only)
public Single fBMR;	Basal Metabolic Rate (in kcal)
public Single fBMRkg;	Basal Metabolic Rate per kilogram in (kcal/kg)
public Single fEstAvg;	Estimated Average Requirement (in kcal)
public Single fBMI;	Body Mass Index
public Single fBFMI;	Body Fat Mass Index (MDD/Quad only)
public Single fFFMI;	Fat Free Mass Index (MDD/Quad only)
public Single fWaistHip;	Waist/Hip ratio
public Single fWellness;	Wellness Marker (TM) (MDD only)
public Single fECW_Legacy;	Legacy ECW calculation (QuanScan mode)
public Single fOHY;	Over hydration
public Single fSkMuscle;	Skeletal muscle mass
public Single fCm;	Cell membrane capacitance
public Single fRext;	R extracellular
public Single fRint;	R intracellular
public Single fFc;	Characteristic frequency
public Single fAlpha;	Alpha angle

1.3.1.1.1.5 BodystatAPI.ESMeasurement Structure

ESMeasurement: Measurement structure. Each test measurement recorded in the device is stored one of these structures. Holds the user inputted information about the subject, together with bio-impedance readings measured by the device (raw electrical readings only, not full results).

C++

```
[System.Runtime.InteropServices.StructLayoutAttribute(System.Runtime.InteropServices.LayoutK
ind.Sequential)]
struct ESMeasurement {
  BSDeviceModel iDeviceModel;
  UInteger ulDeviceSerialNumber;
  Int64 llTestDate;
  Integer iHeight;
  Integer iBCScore;
  Integer iZ_5kHz;
  Integer iZ_16kHz;
  Integer iZ_24kHz;
  Integer iZ_50kHz;
  Integer iZ_140kHz;
  Integer iZ_200kHz;
  Integer iZ_280kHz;
  Single fPA_50kHz;
};
```

C#

```
[System.Runtime.InteropServices.StructLayoutAttribute(System.Runtime.InteropServices.LayoutK
ind.Sequential)]
public struct ESMeasurement {
  public BSDeviceModel iDeviceModel;
  public UInteger ulDeviceSerialNumber;
  public Int64 llTestDate;
  public Integer iHeight;
  public Integer iBCScore;
  public Integer iZ_5kHz;
  public Integer iZ_16kHz;
  public Integer iZ_24kHz;
  public Integer iZ_50kHz;
  public Integer iZ_140kHz;
  public Integer iZ_200kHz;
  public Integer iZ_280kHz;
  public Single fPA_50kHz;
}
```

Visual Basic

```
<System.Runtime.InteropServices.StructLayoutAttribute(System.Runtime.InteropServices.LayoutK
ind.Sequential)>
Public Structure ESMeasurement
  Public iDeviceModel As BSDeviceModel
  Public ulDeviceSerialNumber As UInteger
  Public llTestDate As Int64
  Public iHeight As Integer
  Public iBCScore As Integer
  Public iZ_5kHz As Integer
  Public iZ_16kHz As Integer
  Public iZ_24kHz As Integer
  Public iZ_50kHz As Integer
  Public iZ_140kHz As Integer
  Public iZ_200kHz As Integer
  Public iZ_280kHz As Integer
  Public fPA_50kHz As Single
End Structure
```

Java

```
[System.Runtime.InteropServices.StructLayoutAttribute(System.Runtime.InteropServices.LayoutK
ind.Sequential)]
public class ESMeasurement;
```

MATLAB

```
ESMeasurement
```

File

BodystatSDK.vb

ESMeasurement Fields

	Name	Description
◆	fPA_50kHz (see page 103)	Phase angle measured at 50 kHz (result in degrees) (ES Pro only)
◆	iBCScore (see page 103)	Inputted BC score
◆	iDeviceModel (see page 104)	Device model which performed the measurement
◆	iHeight (see page 104)	Inputted height of the subject (centimeters)
◆	iZ_140kHz (see page 104)	Impedance measured at 140 kHz (result in ohms)
◆	iZ_16kHz (see page 105)	Impedance measured at 16 kHz (result in ohms)
◆	iZ_200kHz (see page 105)	Impedance measured at 200 kHz (result in ohms)
◆	iZ_24kHz (see page 105)	Impedance measured at 24 kHz (result in ohms)
◆	iZ_280kHz (see page 106)	Impedance measured at 280 kHz (result in ohms)
◆	iZ_50kHz (see page 106)	Impedance measured at 50 kHz (result in ohms)
◆	iZ_5kHz (see page 106)	Impedance measured at 5 kHz (result in ohms)
◆	llTestDate (see page 107)	Date & time the test measurement was performed (Windows 64-bit time structure) (MDD/Quad only)
◆	ulDeviceSerialNumber (see page 107)	Unique serial number of the device which performed the measurement

ESMeasurement Properties

	Name	Description
▨	tTestDate (see page 107)	Convert underlying 64-bit time structure to/from Visual Basic Date/Time

1

1.3.1.1.1.5.1 ESMeasurement Fields

1.3.1.1.1.5.1.1 BodystatAPI.ESMeasurement.fPA_50kHz Field

C++

```
Single fPA_50kHz;
```

C#

```
public Single fPA_50kHz;
```

Visual Basic

```
Public fPA_50kHz As Single
```

Java

```
public Single fPA_50kHz;
```

MATLAB

```
fPA_50kHz
```

Description

Phase angle measured at 50 kHz (result in degrees) (ES Pro only)

1.3.1.1.1.5.1.2 BodystatAPI.ESMeasurement.iBCScore Field

C++

```
Integer iBCScore;
```

C#

```
public Integer iBCScore;
```

Visual Basic

```
Public iBCScore As Integer
```

Java

```
public Integer iBCScore;
```

MATLAB

```
iBCScore
```

Description

Inputted BC score

1.3.1.1.1.5.1.3 BodystatAPI.ESMeasurement.iDeviceModel Field

C++

```
BSDeviceModel iDeviceModel;
```

C#

```
public BSDeviceModel iDeviceModel;
```

Visual Basic

```
Public iDeviceModel As BSDeviceModel
```

Java

```
public BSDeviceModel iDeviceModel;
```

MATLAB

```
iDeviceModel
```

Description

Device model which performed the measurement

1.3.1.1.1.5.1.4 BodystatAPI.ESMeasurement.iHeight Field

C++

```
Integer iHeight;
```

C#

```
public Integer iHeight;
```

Visual Basic

```
Public iHeight As Integer
```

Java

```
public Integer iHeight;
```

MATLAB

```
iHeight
```

Description

Inputted height of the subject (centimeters)

1.3.1.1.1.5.1.5 BodystatAPI.ESMeasurement.iZ_140kHz Field

C++

```
Integer iZ_140kHz;
```

C#

```
public Integer iZ_140kHz;
```

Visual Basic

```
Public iZ_140kHz As Integer
```

Java

```
public Integer iZ_140kHz;
```

MATLAB

```
iZ_140kHz
```

Description

Impedance measured at 140 kHz (result in ohms)

1.3.1.1.1.5.1.6 BodystatAPI.ESMeasurement.iZ_16kHz Field

C++

```
Integer iZ_16kHz;
```

C#

```
public Integer iZ_16kHz;
```

Visual Basic

```
Public iZ_16kHz As Integer
```

Java

```
public Integer iZ_16kHz;
```

MATLAB

```
iZ_16kHz
```

Description

Impedance measured at 16 kHz (result in ohms)

1.3.1.1.1.5.1.7 BodystatAPI.ESMeasurement.iZ_200kHz Field

C++

```
Integer iZ_200kHz;
```

C#

```
public Integer iZ_200kHz;
```

Visual Basic

```
Public iZ_200kHz As Integer
```

Java

```
public Integer iZ_200kHz;
```

MATLAB

```
iZ_200kHz
```

Description

Impedance measured at 200 kHz (result in ohms)

1.3.1.1.1.5.1.8 BodystatAPI.ESMeasurement.iZ_24kHz Field

C++

```
Integer iZ_24kHz;
```

C#

```
public Integer iZ_24kHz;
```

Visual Basic

```
Public iZ_24kHz As Integer
```

Java

```
public Integer iZ_24kHz;
```

MATLAB

```
iZ_24kHz
```

Description

Impedance measured at 24 kHz (result in ohms)

1.3.1.1.1.5.1.9 BodystatAPI.ESMeasurement.iZ_280kHz Field

C++

```
Integer iZ_280kHz;
```

C#

```
public Integer iZ_280kHz;
```

Visual Basic

```
Public iZ_280kHz As Integer
```

Java

```
public Integer iZ_280kHz;
```

MATLAB

```
iZ_280kHz
```

Description

Impedance measured at 280 kHz (result in ohms)

1.3.1.1.1.5.1.10 BodystatAPI.ESMeasurement.iZ_50kHz Field

C++

```
Integer iZ_50kHz;
```

C#

```
public Integer iZ_50kHz;
```

Visual Basic

```
Public iZ_50kHz As Integer
```

Java

```
public Integer iZ_50kHz;
```

MATLAB

```
iZ_50kHz
```

Description

Impedance measured at 50 kHz (result in ohms)

1.3.1.1.1.5.1.11 BodystatAPI.ESMeasurement.iZ_5kHz Field

C++

```
Integer iZ_5kHz;
```

C#

```
public Integer iZ_5kHz;
```

Visual Basic

```
Public iZ_5kHz As Integer
```

Java

```
public Integer iZ_5kHz;
```

MATLAB

```
iZ_5kHz
```

Description

Impedance measured at 5 kHz (result in ohms)

1.3.1.1.1.5.1.12 BodystatAPI.ESMeasurement.llTestDate Field

C++

```
Int64 llTestDate;
```

C#

```
public Int64 llTestDate;
```

Visual Basic

```
Public llTestDate As Int64
```

Java

```
public Int64 llTestDate;
```

MATLAB

```
llTestDate
```

Description

Date & time the test measurement was performed (Windows 64-bit time structure) (MDD/Quad only)

1.3.1.1.1.5.1.13 BodystatAPI.ESMeasurement.ulDeviceSerialNumber Field

C++

```
UInteger ulDeviceSerialNumber;
```

C#

```
public UInteger ulDeviceSerialNumber;
```

Visual Basic

```
Public ulDeviceSerialNumber As UInteger
```

Java

```
public UInteger ulDeviceSerialNumber;
```

MATLAB

```
ulDeviceSerialNumber
```

Description

Unique serial number of the device which performed the measurement

1.3.1.1.1.5.2 ESMeasurement Properties

1.3.1.1.1.5.2.1 BodystatAPI.ESMeasurement.tTestDate Property

C++

```
__property DateTime tTestDate;
```

C#

```
public DateTime tTestDate;
```

Visual Basic

```
Public Property tTestDate() As DateTime
```

Java

```
tTestDate
```

MATLAB

```
tTestDate
```

Description

Convert underlying 64-bit time structure to/from Visual Basic Date/Time

1.3.1.1.1.6 BodystatAPI.BodystatAPI.ESNormals Structure

ESNormals: Equistat Normals structure. Holds the normal values or normal ranges for any given test measurement.

C++

```
[StructLayout(LayoutKind.Sequential)]
struct ESNormals {
  Integer iFatKg_L;
  Integer iFatKg_H;
  Integer iFatKg_PBMPerc_L;
  Integer iFatKg_PBMPerc_H;
  Integer iTFV_L;
  Integer iTFV_H;
  Integer iTFVPerc_L;
  Integer iTFVPerc_H;
  Integer iECFV_L;
  Integer iECFV_H;
  Integer iECFV_PBM_L;
  Integer iECFV_PBM_H;
  Integer iECFV_TFVPerc_L;
  Integer iECFV_TFVPerc_H;
  Integer iICFV_L;
  Integer iICFV_H;
  Integer iICFV_PBM_L;
  Integer iICFV_PBM_H;
  Integer iICFV_TFVPerc_L;
  Integer iICFV_TFVPerc_H;
  Integer iPV_L;
  Integer iPV_H;
  Integer iPV_ECFVPerc_L;
  Integer iPV_ECFVPerc_H;
};
```

C#

```
[StructLayout(LayoutKind.Sequential)]
public struct ESNormals {
  public Integer iFatKg_L;
  public Integer iFatKg_H;
  public Integer iFatKg_PBMPerc_L;
  public Integer iFatKg_PBMPerc_H;
  public Integer iTFV_L;
  public Integer iTFV_H;
  public Integer iTFVPerc_L;
  public Integer iTFVPerc_H;
  public Integer iECFV_L;
  public Integer iECFV_H;
  public Integer iECFV_PBM_L;
  public Integer iECFV_PBM_H;
  public Integer iECFV_TFVPerc_L;
  public Integer iECFV_TFVPerc_H;
  public Integer iICFV_L;
  public Integer iICFV_H;
  public Integer iICFV_PBM_L;
  public Integer iICFV_PBM_H;
  public Integer iICFV_TFVPerc_L;
  public Integer iICFV_TFVPerc_H;
  public Integer iPV_L;
  public Integer iPV_H;
  public Integer iPV_ECFVPerc_L;
  public Integer iPV_ECFVPerc_H;
}
```

Visual Basic

```
<StructLayout(LayoutKind.Sequential)>
Public Structure ESNormals
  Public iFatKg_L As Integer
  Public iFatKg_H As Integer
  Public iFatKg_PBMPerc_L As Integer
  Public iFatKg_PBMPerc_H As Integer
  Public iTFV_L As Integer
  Public iTFV_H As Integer
  Public iTFVPerc_L As Integer
  Public iTFVPerc_H As Integer
  Public iECFV_L As Integer
  Public iECFV_H As Integer
  Public iECFV_PBM_L As Integer
  Public iECFV_PBM_H As Integer
  Public iECFV_TFVPerc_L As Integer
  Public iECFV_TFVPerc_H As Integer
  Public iICFV_L As Integer
  Public iICFV_H As Integer
  Public iICFV_PBM_L As Integer
  Public iICFV_PBM_H As Integer
  Public iICFV_TFVPerc_L As Integer
  Public iICFV_TFVPerc_H As Integer
  Public iPV_L As Integer
  Public iPV_H As Integer
  Public iPV_ECFVPerc_L As Integer
  Public iPV_ECFVPerc_H As Integer
End Structure
```

Java

```
[StructLayout(LayoutKind.Sequential)]
public class ESNormals;
```

MATLAB

```
ESNormals
```

File

BodystatSDK.vb

1.3.1.1.1.7 BodystatAPI.ESRawData Structure

ESRawData: Equistat Raw Data structure. Used when downloading test measurements from the device. Contains a fixed array of measurement records together with information about the number of records occupied within the array and test record (see page 112) starting number.

C++

```
[StructLayout(LayoutKind.Sequential)]
struct ESRawData {
  Integer iRecordArraySize;
  [System.Runtime.InteropServices.MarshalAsAttribute(System.Runtime.InteropServices.Unmanage
dType.ByValArray,
SizeConst:=ES_RAWDATA_ARRAYSIZE,
ArraySubType:=System.Runtime.InteropServices.UnmanagedType.Struct)]
ESMeasurement record;
  Integer iTotalNumRecs;
  Integer ulFirstTestNum;
};
```

C#

```
[StructLayout(LayoutKind.Sequential)]
public struct ESRawData {
  public Integer iRecordArraySize;
  [System.Runtime.InteropServices.MarshalAsAttribute(System.Runtime.InteropServices.Unmanage
dType.ByValArray,
SizeConst:=ES_RAWDATA_ARRAYSIZE,
ArraySubType:=System.Runtime.InteropServices.UnmanagedType.Struct)]
public ESMeasurement() record;
  public Integer iTotalNumRecs;
  public Integer ulFirstTestNum;
}
```

Visual Basic

```
  <StructLayout(LayoutKind.Sequential)>
  Public Structure ESRawData
    Public iRecordArraySize As Integer
    <System.Runtime.InteropServices.MarshalAsAttribute(System.Runtime.InteropServices.Unmanage
  dType.ByValArray,
  SizeConst:=ES_RAWDATA_ARRAYSIZE,
  ArraySubType:=System.Runtime.InteropServices.UnmanagedType.Struct)>
  Public record As ESMeasurement()
    Public iTotalNumRecs As Integer
    Public ulFirstTestNum As Integer
  End Structure
```

Java

```
  [StructLayout(LayoutKind.Sequential)]
  public class ESRawData;
```

MATLAB

```
  ESRawData
```

File

BodystatSDK.vb

Remarks

Most models index 1000 records and store the last 100 at any given time.

Methods

	Name	Description
	New (see page 110)	Override the constructor to initialise required variables in the structure. You must ensure you allocate the structure in a manner which calls this function.

ESRawData Fields

	Name	Description
◆	iRecordArraySize (see page 111)	Maximum number of records this structure has reserved space for (currently fixed)
◆	iTotalNumRecs (see page 111)	Number of populated records in the array
◆	record (see page 112)	Array of test measurement data
◆	ulFirstTestNum (see page 112)	Test number of the first record (see page 112) stored in the array.

1.3.1.1.1.7.1 BodystatAPI.ESRawData.New Constructor

Override the constructor to initialise required variables in the structure. You must ensure you allocate the structure in a manner which calls this function.

C++

```
  ESRawData(ByVal Integer dontcare);
```

C#

```
  public New(ByVal Integer dontcare);
```

Visual Basic

```
  Public Sub New(ByVal dontcare As Integer)
```

Java

```
public ESRawData(ByVal Integer dontcare);
```

MATLAB

```
New
```

Parameters

Parameters	Description
ByVal dontcare As Integer	Ignored: VB requires New() constructors to specify a non-optional parameter for structures. The parameter is not required by the wrapper and is ignored.

Body Source

```
Public Sub New(ByVal dontcare As Integer)
    Me.iRecordArraySize = ES_RAWDATA_ARRAYSIZE  ' Value must be populated with size of
array (currently fixed)
End Sub
```

1.3.1.1.1.7.2 ESRawData Fields

1.3.1.1.1.7.2.1 BodystatAPI.ESRawData.iRecordArraySize Field

C++

```
Integer iRecordArraySize;
```

C#

```
public Integer iRecordArraySize;
```

Visual Basic

```
Public iRecordArraySize As Integer
```

Java

```
public Integer iRecordArraySize;
```

MATLAB

```
iRecordArraySize
```

Description

Maximum number of records this structure has reserved space for (currently fixed)

1.3.1.1.1.7.2.2 BodystatAPI.ESRawData.iTotalNumRecs Field

C++

```
Integer iTotalNumRecs;
```

C#

```
public Integer iTotalNumRecs;
```

Visual Basic

```
Public iTotalNumRecs As Integer
```

Java

```
public Integer iTotalNumRecs;
```

MATLAB

```
iTotalNumRecs
```

Description

Number of populated records in the array

1.3.1.1.1.7.2.3 BodystatAPI.ESRawData.record Field

C++

```
[System.Runtime.InteropServices.MarshalAsAttribute(System.Runtime.InteropServices.UnmanagedT
ype.ByValArray,
SizeConst:=ES_RAWDATA_ARRAYSIZE,
ArraySubType:=System.Runtime.InteropServices.UnmanagedType.Struct)]
ESMeasurement record;
```

C#

```
[System.Runtime.InteropServices.MarshalAsAttribute(System.Runtime.InteropServices.UnmanagedT
ype.ByValArray,
SizeConst:=ES_RAWDATA_ARRAYSIZE,
ArraySubType:=System.Runtime.InteropServices.UnmanagedType.Struct)]
public ESMeasurement() record;
```

Visual Basic

```
<System.Runtime.InteropServices.MarshalAsAttribute(System.Runtime.InteropServices.UnmanagedT
ype.ByValArray,
SizeConst:=ES_RAWDATA_ARRAYSIZE,
ArraySubType:=System.Runtime.InteropServices.UnmanagedType.Struct)>
Public record As ESMeasurement()
```

Java

```
[System.Runtime.InteropServices.MarshalAsAttribute(System.Runtime.InteropServices.UnmanagedT
ype.ByValArray,
SizeConst:=ES_RAWDATA_ARRAYSIZE,
ArraySubType:=System.Runtime.InteropServices.UnmanagedType.Struct)]
public ESMeasurement() record;
```

MATLAB

```
record
```

Description

Array of test measurement data

1.3.1.1.1.7.2.4 BodystatAPI.ESRawData.ulFirstTestNum Field

C++

```
Integer ulFirstTestNum;
```

C#

```
public Integer ulFirstTestNum;
```

Visual Basic

```
Public ulFirstTestNum As Integer
```

Java

```
public Integer ulFirstTestNum;
```

MATLAB

```
ulFirstTestNum
```

Description

Test number of the first record (see page 112) stored in the array.

1.3.1.1.1.8 BodystatAPI.BodystatAPI.ESResults Structure

ESResults: Equistst Results structure. Holds the calculated results for any given test measurement.

C++

```
[StructLayout(LayoutKind.Sequential)]
struct ESResults {
    Single fFatKg;
    Single fPBM;
    Single fIBM;
    Single fFatKg_PBMPerc;
    Single fTFV;
    Single fTFVPerc;
    Single fECFV;
    Single fECFV_PBM;
    Single fECFV_TFVPerc;
    Single fICFV;
    Single fICFV_PBM;
    Single fICFV_TFVPerc;
    Single fPV;
    Single fPV_ECFVPerc;
    Single fDehydration;
  };
```

C#

```
  [StructLayout(LayoutKind.Sequential)]
  public struct ESResults {
    public Single fFatKg;
    public Single fPBM;
    public Single fIBM;
    public Single fFatKg_PBMPerc;
    public Single fTFV;
    public Single fTFVPerc;
    public Single fECFV;
    public Single fECFV_PBM;
    public Single fECFV_TFVPerc;
    public Single fICFV;
    public Single fICFV_PBM;
    public Single fICFV_TFVPerc;
    public Single fPV;
    public Single fPV_ECFVPerc;
    public Single fDehydration;
  }
```

Visual Basic

```
  <StructLayout(LayoutKind.Sequential)>
  Public Structure ESResults
    Public fFatKg As Single
    Public fPBM As Single
    Public fIBM As Single
    Public fFatKg_PBMPerc As Single
    Public fTFV As Single
    Public fTFVPerc As Single
    Public fECFV As Single
    Public fECFV_PBM As Single
    Public fECFV_TFVPerc As Single
    Public fICFV As Single
    Public fICFV_PBM As Single
    Public fICFV_TFVPerc As Single
    Public fPV As Single
    Public fPV_ECFVPerc As Single
    Public fDehydration As Single
  End Structure
```

Java

```
  [StructLayout(LayoutKind.Sequential)]
  public class ESResults;
```

MATLAB

```
  ESResults
```

File

BodystatSDK.vb

1.3.1.1.1.9 BodystatAPI.BodystatAPI.ImpData Structure

Represents one frequency measurement

C++

```
[StructLayout(LayoutKind.Sequential)]
struct ImpData {
  Single fFrequency;
  Single fImpedance;
  Single fPhaseAngle;
};
```

C#

```
[StructLayout(LayoutKind.Sequential)]
public struct ImpData {
  public Single fFrequency;
  public Single fImpedance;
  public Single fPhaseAngle;
}
```

Visual Basic

```
<StructLayout(LayoutKind.Sequential)>
Public Structure ImpData
  Public fFrequency As Single
  Public fImpedance As Single
  Public fPhaseAngle As Single
End Structure
```

Java

```
[StructLayout(LayoutKind.Sequential)]
public class ImpData;
```

MATLAB

```
ImpData
```

File

BodystatSDK.vb

Members

Members	Description
public Single fFrequency;	Frequency in kHz
public Single fImpedance;	Absolute value of impedance measured at given frequency (result in ohms)
public Single fPhaseAngle;	Phase angle (result in degrees)

1.3.1.1.2 BodystatAPI Fields

1.3.1.1.2.1 BodystatAPI.BS_RAWDATA_ARRAYSIZE Field

C++

```
Integer BS_RAWDATA_ARRAYSIZE = 100;
```

C#

```
public const Integer BS_RAWDATA_ARRAYSIZE = 100;
```

Visual Basic

```
Public Const BS_RAWDATA_ARRAYSIZE As Integer = 100
```

Java

```
public Integer BS_RAWDATA_ARRAYSIZE = 100;
```

MATLAB

```
BS_RAWDATA_ARRAYSIZE
```

Description

Number of measurement records BSRawData (see page 96) structure should reserved space for (currently fixed)

1.3.1.1.2.2 BodystatAPI.ES_RAWDATA_ARRAYSIZE Field

C++

```
Integer ES_RAWDATA_ARRAYSIZE = 100;
```

C#

```
public const Integer ES_RAWDATA_ARRAYSIZE = 100;
```

Visual Basic

```
Public Const ES_RAWDATA_ARRAYSIZE As Integer = 100
```

Java

```
public Integer ES_RAWDATA_ARRAYSIZE = 100;
```

MATLAB

```
ES_RAWDATA_ARRAYSIZE
```

Description

Number of measurement records ESRawData (see page 109) structure should reserved space for (currently fixed)

1

1.3.1.1.3 BodystatAPI Methods

1.3.1.1.3.1 BodystatAPI.BSAuthenticateBTDevice Method

Authenticate (pair) a Bodystat device found during a previous call to BSSearchBTDevices (see page 136). Completely silent implementation.

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
Integer BSAuthenticateBTDevice(ByVal IntPtr hWndParent, Optional ByVal UInt16 iTimeout = 7
_);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared Integer BSAuthenticateBTDevice(ByVal IntPtr hWndParent, Optional ByVal UInt16
iTimeout);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function BSAuthenticateBTDevice(ByVal hWndParent As IntPtr, Optional ByVal
iTimeout As UInt16 = 7 _) As Integer
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared Integer BSAuthenticateBTDevice(ByVal IntPtr hWndParent, Optional ByVal UInt16
iTimeout);
```

MATLAB

```
BSAuthenticateBTDevice
```

Parameters

Parameters	Description
ByVal hWndParent As IntPtr	Handle of the parent window.
Optional ByVal iTimeout As UInt16 = 7 _	Timeout multiplier supplied to the Bluetooth radio search API. Controls the amount of time to spend searching for the device. This function will block for this duration so is better called from a thread. Each multiplier increment equates to about 1.28 seconds. Thus a timeout value of 10 = 12.8s. Maximum time permitted by the API is a multiplier of 45 (just under 1 minute), values higher than this will be capped at 45. Default value is 7 (about 10 seconds).

Returns

TRUE if it succeeds, FALSE if it fails.

Remarks

Note this process can be time consuming to complete and executes synchronously. Either call from a threaded function or illustrate operation in progress to the user e.g. by means of an indefinite progress bar or busy mouse cursor, until the task completes or times out.

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function BSAuthenticateBTDevice( _
ByVal hWndParent As IntPtr _
, Optional ByVal iTimeout As UInt16 = 7 _
) As Integer
End Function
```

1.3.1.1.3.2 BodystatAPI.BSAutoSetupBT Method

Automatic Bluetooth setup. Automatically searches available Bluetooth devices, exchanges passkeys with Bodystat device and enables serial port service required for communication with the device. If multiple Bodystat devices are detected the user is prompted to chose the desired device (model and BT BDA address shown).

This function is typically called as a one time setup for the device (e.g. after first installation, or from the hardware configuration page of your application's settings, or when the user obtains a replacement device (upgrades/replacement/etc).

Typically, once setup you need only store the device com port returned. You can then subsequently communicate with the device knowing only the com port. It is not necessary to setup the device in this way before each use.

New non-paired units will only be found if switched on, in range and ready for communication (on main startup screen). It is advisable to instruct the user to ensure this is the case before calling.

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
Integer BSAutoSetupBT(ByVal StringBuilder lpComPort, ByVal Integer iComPortBufferSize,
ByVal Integer bReportErrors, ByVal IntPtr hWndParent);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared Integer BSAutoSetupBT(ByVal StringBuilder lpComPort, ByVal Integer
iComPortBufferSize, ByVal Integer bReportErrors, ByVal IntPtr hWndParent);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function BSAutoSetupBT(ByVal lpComPort As StringBuilder, ByVal
iComPortBufferSize As Integer, ByVal bReportErrors As Integer, ByVal hWndParent As IntPtr)
As Integer
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared Integer BSAutoSetupBT(ByVal StringBuilder lpComPort, ByVal Integer
iComPortBufferSize, ByVal Integer bReportErrors, ByVal IntPtr hWndParent);
```

MATLAB

```
BSAutoSetupBT
```

Parameters

Parameters	Description
ByVal lpComPort As StringBuilder	[out] String buffer to receive the device port the device is assigned to.
ByVal bReportErrors As Integer	TRUE to report errors to the user (shown as message boxes, locale constraints apply). FALSE to hide errors from the user (still not completely silent).
ByVal hWndParent As IntPtr	Handle of the parent window.
iSize	The size of the buffer.

Returns

TRUE if it succeeds, FALSE if it fails.

Remarks

Note this process can be time consuming to complete and executes synchronously. Either call from a threaded function or illustrate operation in progress to the user e.g. by means of an indefinite progress bar or busy mouse cursor, until the task completes or timesout.

Errors during the process can be optionally shown/hidden. However, a completely silent approach is not possible using this function (instead use BSSearchBTDevices (see page 136) and BSGetBTBodystatDevice (see page 120) or iterate radio handles manually. Then pass desired handle to BSAuthenticateBTDevice (see page 115) for pairing).

Finally be aware this function is only available for supported locales. Otherwise English messages may be displayed to the user. Adopt silent approach discussed previously if you are targeting a non-supported locale.

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function BSAutoSetupBT( _
ByVal lpComPort As StringBuilder _
, ByVal iComPortBufferSize As Integer _
, ByVal bReportErrors As Integer _
, ByVal hWndParent As IntPtr _
) As Integer
End Function
```

1.3.1.1.3.3 BodystatAPI.BSCalculateNormals Method

Calculate normal values and normal ranges for a given subject and their bio-impedance measurement together with the matching results structure for that measurement. You must supply a populated results structure which pertains to the measurement being supplied (obtained from BSCalculateResults (see page 118)).

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
BSError BSCalculateNormals(ByRef BSMeasurement pM, ByRef BSResults pR, ByRef BSNormals pN);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSCalculateNormals(ByRef BSMeasurement pM, ByRef BSResults pR, ByRef
BSNormals pN);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function BSCalculateNormals(ByRef pM As BSMeasurement, ByRef pR As BSResults,
ByRef pN As BSNormals) As BSError
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSCalculateNormals(ByRef BSMeasurement pM, ByRef BSResults pR, ByRef
BSNormals pN);
```

MATLAB

```
BSCalculateNormals
```

Parameters

Parameters	Description
ByRef pM As BSMeasurement	The measurement details downloaded from the device (includes model, user inputted parameters and bio-impedance measurement readings).
ByRef pR As BSResults	The results calculated for this measurement previously.
ByRef pN As BSNormals	[out] Receives the normal values and normal ranges for this measurement.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

Normals vary by subject attributes, measured bio-impedance values and calculated results.

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function BSCalculateNormals( _
ByRef pM As BSMeasurement _
, ByRef pR As BSResults _
, ByRef pN As BSNormals _
) As BSError
End Function
```

1.3.1.1.3.4 BodystatAPI.BSCalculateResults Method

Calculate complete result set for a given bio-impedance measurement recorded by the Bodystat device.

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
BSError BSCalculateResults(ByRef BSMeasurement pM, ByRef BSResults pR, ByVal BSAnalysisMode
iAnalysisMode);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSCalculateResults(ByRef BSMeasurement pM, ByRef BSResults pR, ByVal
BSAnalysisMode iAnalysisMode);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function BSCalculateResults(ByRef pM As BSMeasurement, ByRef pR As BSResults,
ByVal iAnalysisMode As BSAnalysisMode) As BSError
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSCalculateResults(ByRef BSMeasurement pM, ByRef BSResults pR, ByVal
BSAnalysisMode iAnalysisMode);
```

MATLAB

```
BSCalculateResults
```

Parameters

Parameters	Description
ByRef pM As BSMeasurement	The measurement details downloaded from the device (includes model, user inputted parameters and bio-impedance measurement readings).
ByRef pR As BSResults	[out] Receives the complete calculated results for this measurement.
ByVal iAnalysisMode As BSAnalysisMode	The desired analysis mode (see BSAnalysisMode (see page 145) for more information).

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

Results vary by subject attributes and measured bio-impedance values.

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function BSCalculateResults( _
ByRef pM As BSMeasurement _
, ByRef pR As BSResults _
, ByVal iAnalysisMode As BSAnalysisMode _
) As BSError
End Function
```

1.3.1.1.3.5 BodystatAPI.BSCloseComport Method

Close com port.

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
BSCloseComport(_);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSCloseComport(_);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Sub BSCloseComport( As _)
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSCloseComport(_ );
```

MATLAB

```
BSCloseComport
```

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Sub BSCloseComport( _
)
End Sub
```

1.3.1.1.3.6 BodystatAPI.BSConnect Method

Connect to the Bodystat device and ensure the device is responding.

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
BSError BSConnect(_);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSConnect(_);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function BSConnect( As _) As BSError
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSConnect(_ );
```

MATLAB

```
BSConnect
```

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have opened the com port by calling BSOpenComPort previously.

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function BSConnect( _
) As BSError
End Function
```

1.3.1.1.3.7 BodystatAPI.BSGetBTBodystatDevice Method

Used to get information about any Bodystat device found after a previous call to BSSearchBTDevices (see page 136).

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
Integer BSGetBTBodystatDevice(ByVal StringBuilder lpDeviceName, ByVal Integer
iDeviceNameBufferSize, ByVal StringBuilder lpBDA, ByVal Integer iBDABufferSize, ByVal
StringBuilder lpComPort, ByVal Integer iComPortBufferSize, Optional ByVal UInt16 iTimeout =
7 _);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared Integer BSGetBTBodystatDevice(ByVal StringBuilder lpDeviceName, ByVal Integer
iDeviceNameBufferSize, ByVal StringBuilder lpBDA, ByVal Integer iBDABufferSize, ByVal
StringBuilder lpComPort, ByVal Integer iComPortBufferSize, Optional ByVal UInt16 iTimeout);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function BSGetBTBodystatDevice(ByVal lpDeviceName As StringBuilder, ByVal
iDeviceNameBufferSize As Integer, ByVal lpBDA As StringBuilder, ByVal iBDABufferSize As
Integer, ByVal lpComPort As StringBuilder, ByVal iComPortBufferSize As Integer, Optional
ByVal iTimeout As UInt16 = 7 _) As Integer
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared Integer BSGetBTBodystatDevice(ByVal StringBuilder lpDeviceName, ByVal Integer
iDeviceNameBufferSize, ByVal StringBuilder lpBDA, ByVal Integer iBDABufferSize, ByVal
StringBuilder lpComPort, ByVal Integer iComPortBufferSize, Optional ByVal UInt16 iTimeout);
```

MATLAB

```
BSGetBTBodystatDevice
```

Parameters

Parameters	Description
ByVal lpDeviceName As StringBuilder	[out] String buffer to receive the product name of the Bodystat device.
ByVal iDeviceNameBufferSize As Integer	Size of the device name buffer.
ByVal lpBDA As StringBuilder	[out] String buffer to receive the Bluetooth hexadecimal address (known as the BDA - unique to each BT module in a device, like a MAC address).
ByVal iBDABufferSize As Integer	Size of the BDA buffer.
ByVal lpComPort As StringBuilder	[out] String buffer to receive device/port the unit is assigned to.
Optional ByVal iTimeout As UInt16 = 7 _	Timeout multiplier supplied to the Bluetooth radio search API. Controls the amount of time to spend searching for the device. This function will block for this duration so is better called from a thread. Each multiplier increment equates to about 1.28 seconds. Thus a timeout value of 10 = 12.8s. Maximum time permitted by the API is a multiplier of 45 (just under 1 minute), values higher than this will be capped at 45. Default value is 7 (about 10 seconds).
iComBufferSize	Size of the com port buffer.

Returns

TRUE if it succeeds, FALSE if it fails.

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function BSGetBTBodystatDevice( _
ByVal lpDeviceName As StringBuilder _
, ByVal iDeviceNameBufferSize As Integer _
, ByVal lpBDA As StringBuilder _
, ByVal iBDABufferSize As Integer _
, ByVal lpComPort As StringBuilder _
, ByVal iComPortBufferSize As Integer _
, Optional ByVal iTimeout As UInt16 = 7 _
) As Integer
End Function
```

1.3.1.1.3.8 BodystatAPI.BSGetBTStackInfo Method

Retrieve information about the Bluetooth stack. Useful for diagnostics.

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
```

```
Integer BSGetBTStackInfo(ByVal StringBuilder lpBuffer, ByVal Integer iBufferSize);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared Integer BSGetBTStackInfo(ByVal StringBuilder lpBuffer, ByVal Integer
iBufferSize);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function BSGetBTStackInfo(ByVal lpBuffer As StringBuilder, ByVal iBufferSize
As Integer) As Integer
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared Integer BSGetBTStackInfo(ByVal StringBuilder lpBuffer, ByVal Integer
iBufferSize);
```

MATLAB

```
BSGetBTStackInfo
```

Parameters

Parameters	Description
ByVal lpBuffer As StringBuilder	[out] String buffer to receive the information.
ByVal iBufferSize As Integer	Size of the buffer.

Returns

true if it succeeds, false if it fails.

Remarks

Note: Information can only be provided for supported BT radios.

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function BSGetBTStackInfo( _
ByVal lpBuffer As StringBuilder _
, ByVal iBufferSize As Integer _
) As Integer
End Function
```

1.3.1.1.3.9 BodystatAPI.BSGetDeviceFamily Method

Determine the device family of a specific model.

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
BSDeviceFamily BSGetDeviceFamily(ByVal BSDeviceModel iModel);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSDeviceFamily BSGetDeviceFamily(ByVal BSDeviceModel iModel);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function BSGetDeviceFamily(ByVal iModel As BSDeviceModel) As BSDeviceFamily
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
```

```
public Shared BSDeviceFamily BSGetDeviceFamily(ByVal BSDeviceModel iModel);
```

MATLAB

```
BSGetDeviceFamily
```

Parameters

Parameters	Description
ByVal iModel As BSDeviceModel	The desired model of interest.

Returns

The device family the model belongs to.

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function BSGetDeviceFamily( _
ByVal iModel As BSDeviceModel _
) As BSDeviceFamily
End Function
```

1.3.1.1.3.10 BodystatAPI.BSGetDeviceModelName Method

Determine the full product name of a specific model.

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
BSError BSGetDeviceModelName(ByVal BSDeviceModel iModel, ByVal StringBuilder lpBuffer,
ByVal Integer iBufferSize);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSGetDeviceModelName(ByVal BSDeviceModel iModel, ByVal StringBuilder
lpBuffer, ByVal Integer iBufferSize);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function BSGetDeviceModelName(ByVal iModel As BSDeviceModel, ByVal lpBuffer
As StringBuilder, ByVal iBufferSize As Integer) As BSError
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSGetDeviceModelName(ByVal BSDeviceModel iModel, ByVal StringBuilder
lpBuffer, ByVal Integer iBufferSize);
```

MATLAB

```
BSGetDeviceModelName
```

Parameters

Parameters	Description
ByVal iModel As BSDeviceModel	The desired model of interest.
ByVal lpBuffer As StringBuilder	[out] String buffer to receive the name.
ByVal iBufferSize As Integer	Size of the buffer.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function BSGetDeviceModelName( _
ByVal iModel As BSDeviceModel _
, ByVal lpBuffer As StringBuilder _
, ByVal iBufferSize As Integer _
) As BSError
End Function
```

1.3.1.1.3.11 BodystatAPI.BSGetDeviceModelNameShort Method

Determine the short product name of a specific model.

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
BSError BSGetDeviceModelNameShort(ByVal BSDeviceModel iModel, ByVal StringBuilder lpBuffer,
ByVal Integer iBufferSize);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSGetDeviceModelNameShort(ByVal BSDeviceModel iModel, ByVal
StringBuilder lpBuffer, ByVal Integer iBufferSize);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function BSGetDeviceModelNameShort(ByVal iModel As BSDeviceModel, ByVal
lpBuffer As StringBuilder, ByVal iBufferSize As Integer) As BSError
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSGetDeviceModelNameShort(ByVal BSDeviceModel iModel, ByVal
StringBuilder lpBuffer, ByVal Integer iBufferSize);
```

MATLAB

```
BSGetDeviceModelNameShort
```

Parameters

Parameters	Description
ByVal iModel As BSDeviceModel	The desired model of interest.
ByVal lpBuffer As StringBuilder	[out] String buffer to receive the name.
ByVal iBufferSize As Integer	Size of the buffer.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function BSGetDeviceModelNameShort( _
ByVal iModel As BSDeviceModel _
, ByVal lpBuffer As StringBuilder _
, ByVal iBufferSize As Integer _
) As BSError
End Function
```

1.3.1.1.3.12 BodystatAPI.BSGetSdkLibraryVersion Method

Determine the Bodystat SDK library (DLL) version.

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
BSError BSGetSdkLibraryVersion(ByRef Integer pMajorSdkVer, ByRef Integer pMinorSdkVer);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSGetSdkLibraryVersion(ByRef Integer pMajorSdkVer, ByRef Integer
pMinorSdkVer);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function BSGetSdkLibraryVersion(ByRef pMajorSdkVer As Integer, ByRef
pMinorSdkVer As Integer) As BSError
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSGetSdkLibraryVersion(ByRef Integer pMajorSdkVer, ByRef Integer
pMinorSdkVer);
```

MATLAB

```
BSGetSdkLibraryVersion
```

Parameters

Parameters	Description
ByRef pMajorSdkVer As Integer	[out] Receives the major version number of the Bodystat SDK DLL.
ByRef pMinorSdkVer As Integer	[out] Receives the minor version number of the Bodystat SDK DLL.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

Caller should check the version number of the DLL if utilising functionality noted only as being available in later revisions of the SDK. Or, alternatively, ensure the SDK DLL is updated appropriately during installation of your application.

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function BSGetSdkLibraryVersion( _
ByRef pMajorSdkVer As Integer _
, ByRef pMinorSdkVer As Integer _
) As BSError
End Function
```

1.3.1.1.3.13 BodystatAPI.BSIsBTAvailable Method

Check if a supported BT radio is available on this computer. Checks for presence of USB Bluetooth dongles and inbuilt Bluetooth modules in the PC. No communication with Bodystat devices takes place during this query.

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
Integer BSIsBTAvailable(_);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared Integer BSIsBTAvailable(_);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function BSIsBTAvailable( As _) As Integer
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared Integer BSIsBTAvailable(_ );
```

MATLAB

```
BSIsBTAvailable
```

Returns

Returns true if supported radio is available and enabled.

Remarks

Note: Only supported BT radios can be queried (presently those compatible with the Microsoft Bluetooth APIs and some Widdcomm models).

Also note, Bluetooth must be enabled on the PC by the user to permit detection. Some PCs, laptops particularly, have physical switches to enable bluetooth/wireless communication.

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function BSIsBTAvailable( _
) As Integer
End Function
```

1.3.1.1.3.14 BodystatAPI.BSOpenComport Method

Open com port to the device ready for communication.

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
BSError BSOpenComport(ByVal StringBuilder lpComPort, ByVal Integer iComPortBufferSize);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSOpenComport(ByVal StringBuilder lpComPort, ByVal Integer
iComPortBufferSize);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function BSOpenComport(ByVal lpComPort As StringBuilder, ByVal
iComPortBufferSize As Integer) As BSError
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSOpenComport(ByVal StringBuilder lpComPort, ByVal Integer
iComPortBufferSize);
```

MATLAB

```
BSOpenComport
```

Parameters

Parameters	Description
ByVal lpComPort As StringBuilder	String buffer containing the device/port to be opened.
iSize	The size of the buffer.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function BSOpenComport( _
ByVal lpComPort As StringBuilder _
, ByVal iComPortBufferSize As Integer _
) As BSError
End Function
```

1.3.1.1.3.15 BSReadCalibrationTime Method

1.3.1.1.3.15.1 BodystatAPI.BSReadCalibrationTime Method (DateTime)

Read the calibration date from the connected device (when the unit was last calibrated). Only the date element of this value should be shown. Exact time value is not relevant.

C++

```
BSError BSReadCalibrationTime(ByRef DateTime dtCalibrationTime);
```

C#

```
public Shared BSError BSReadCalibrationTime(ByRef DateTime dtCalibrationTime);
```

Visual Basic

```
Public Shared Function BSReadCalibrationTime(ByRef dtCalibrationTime As DateTime) As BSError
```

Java

```
public Shared BSError BSReadCalibrationTime(ByRef DateTime dtCalibrationTime);
```

MATLAB

```
BSReadCalibrationTime
```

Parameters

Parameters	Description
ByRef dtCalibrationTime As DateTime	[out] Receives the date the device was last calibrated.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect (see page 120) previously.

Body Source

```
Public Shared Function BSReadCalibrationTime( _
    ByRef dtCalibrationTime As DateTime _
    ) As BSError

    Dim llCalibrationDateTime As Int64
    Dim bsErr As BSError = BSReadCalibrationTime(llCalibrationDateTime)

    dtCalibrationTime = "#1/1/1970#"
    dtCalibrationTime = dtCalibrationTime.AddSeconds(llCalibrationDateTime)

    Return bsErr
End Function
```

1.3.1.1.3.15.2 BodystatAPI.BSReadCalibrationTime Method (Int64)

Read the calibration date from the connected device (when the unit was last calibrated). Only the date element of this value should be shown. Exact time value is not relevant.

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
```

```
BSError BSReadCalibrationTime(ByRef Int64 pCalibrationTime);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSReadCalibrationTime(ByRef Int64 pCalibrationTime);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function BSReadCalibrationTime(ByRef pCalibrationTime As Int64) As BSError
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSReadCalibrationTime(ByRef Int64 pCalibrationTime);
```

MATLAB

```
BSReadCalibrationTime
```

Parameters

Parameters	Description
ByRef pCalibrationTime As Int64	[out] Receives the date the device was last calibrated.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect (see page 120) previously.

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function BSReadCalibrationTime( _
ByRef pCalibrationTime As Int64 _
) As BSError
End Function
```

1.3.1.1.3.16 BSReadCurrentTime Method

1.3.1.1.3.16.1 BodystatAPI.BSReadCurrentTime Method (DateTime, Integer)

Read the current time from the internal clock of the connected device. Note not all device models support this function (not all have an internal clock).

C++

```
BSError BSReadCurrentTime(ByRef DateTime dtCurrentTime, Optional ByVal Integer iDST = -1 _);
```

C#

```
public Shared BSError BSReadCurrentTime(ByRef DateTime dtCurrentTime, Optional ByVal
Integer iDST);
```

Visual Basic

```
Public Shared Function BSReadCurrentTime(ByRef dtCurrentTime As DateTime, Optional ByVal
iDST As Integer = -1 _) As BSError
```

Java

```
public Shared BSError BSReadCurrentTime(ByRef DateTime dtCurrentTime, Optional ByVal
Integer iDST);
```

MATLAB

```
BSReadCurrentTime
```

Parameters

Parameters	Description
ByRef dtCurrentTime As DateTime	[out] Receives the current time of the device's internal clock.
Optional ByVal iDST As Integer = -1 _	Daylight Saving Time handling (as per Microsoft time functions) Zero (0) to indicate that standard time is in effect. A value greater than 0 to indicate that daylight saving time is in effect. A value less than zero to have the C run-time library code compute whether standard time or daylight saving time is in effect. (default)

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect (see page 120) previously.

Body Source

```
Public Shared Function BSReadCurrentTime( _
    ByRef dtCurrentTime As DateTime _
    , Optional ByVal iDST As Integer = -1 _
    ) As BSError

    Dim llCurrentDateTime As Int64
    Dim bsErr As BSError = BSReadCurrentTime(llCurrentDateTime, iDST)

    dtCurrentTime = "#1/1/1970#"
    dtCurrentTime = dtCurrentTime.AddSeconds(llCurrentDateTime)
    If (iDST <> 0) Then
        dtCurrentTime = dtCurrentTime.ToLocalTime()
    End If

    Return bsErr
End Function
```

1.3.1.1.3.16.2 BodystatAPI.BSReadCurrentTime Method (Int64, Integer)

Read the current time from the internal clock of the connected device. Note not all device models support this function (not all have an internal clock).

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
BSError BSReadCurrentTime(ByRef Int64 pCurrentTime, Optional ByVal Integer iDST = -1 _);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSReadCurrentTime(ByRef Int64 pCurrentTime, Optional ByVal Integer
iDST);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function BSReadCurrentTime(ByRef pCurrentTime As Int64, Optional ByVal iDST
As Integer = -1 _) As BSError
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSReadCurrentTime(ByRef Int64 pCurrentTime, Optional ByVal Integer
iDST);
```

MATLAB

```
BSReadCurrentTime
```

Parameters

Parameters	Description
ByRef pCurrentTime As Int64	[out] Receives the current time of the device's internal clock.
Optional ByVal iDST As Integer = -1 _	Daylight Saving Time handling (as per Microsoft time functions) Zero (0) to indicate that standard time is in effect. A value greater than 0 to indicate that daylight saving time is in effect. A value less than zero to have the C run-time library code compute whether standard time or daylight saving time is in effect. (default)

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect (see page 120) previously.

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function BSReadCurrentTime( _
ByRef pCurrentTime As Int64 _
, Optional ByVal iDST As Integer = -1 _
) As BSError
End Function
```

1.3.1.1.3.17 BodystatAPI.BSReadModelVersion Method

Read device model and firmware version of the connected Bodystat device.

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
BSError BSReadModelVersion(ByRef BSDeviceModel pModel, ByRef Byte pMajorVersion, ByRef Byte
pMinorVersion, ByRef Byte pPsoc2Version, ByRef Byte pEepromVersion, ByRef Byte
pBluetoothInfo);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSReadModelVersion(ByRef BSDeviceModel pModel, ByRef Byte
pMajorVersion, ByRef Byte pMinorVersion, ByRef Byte pPsoc2Version, ByRef Byte
pEepromVersion, ByRef Byte pBluetoothInfo);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function BSReadModelVersion(ByRef pModel As BSDeviceModel, ByRef
pMajorVersion As Byte, ByRef pMinorVersion As Byte, ByRef pPsoc2Version As Byte, ByRef
pEepromVersion As Byte, ByRef pBluetoothInfo As Byte) As BSError
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSReadModelVersion(ByRef BSDeviceModel pModel, ByRef Byte
pMajorVersion, ByRef Byte pMinorVersion, ByRef Byte pPsoc2Version, ByRef Byte
pEepromVersion, ByRef Byte pBluetoothInfo);
```

MATLAB

```
BSReadModelVersion
```

Parameters

Parameters	Description
ByRef pModel As BSDeviceModel	[out] Receives the device model.

ByRef pMajorVersion As Byte	[out] Receives the major version of the main firmware in the device.
ByRef pMinorVersion As Byte	[out] Receives the minor version of the main firmware in the device.
ByRef pPsoc2Version As Byte	[out] Receives the version of Psoc2 firmware in the device.
ByRef pEepromVersion As Byte	[out] Receives the version of the EEprom firmware in the device.
ByRef pBluetoothInfo As Byte	[out] Receives the version of the Bluetooth module in the device.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect (see page 120) previously.

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function BSReadModelVersion( _
ByRef pModel As BSDeviceModel _
, ByRef pMajorVersion As Byte _
, ByRef pMinorVersion As Byte _
, ByRef pPsoc2Version As Byte _
, ByRef pEepromVersion As Byte _
, ByRef pBluetoothInfo As Byte _
) As BSError
End Function
```

1.3.1.1.3.18 BodystatAPI.BSReadPrinterAddress Method

Read the printer address the connected device is paired with (for direct printing). Not all device models support this function (not all support direct printing).

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
BSError BSReadPrinterAddress(ByVal StringBuilder lpPrinterAddress, ByVal Integer
iPrinterAddressBufferSize);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSReadPrinterAddress(ByVal StringBuilder lpPrinterAddress, ByVal
Integer iPrinterAddressBufferSize);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function BSReadPrinterAddress(ByVal lpPrinterAddress As StringBuilder, ByVal
iPrinterAddressBufferSize As Integer) As BSError
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSReadPrinterAddress(ByVal StringBuilder lpPrinterAddress, ByVal
Integer iPrinterAddressBufferSize);
```

MATLAB

```
BSReadPrinterAddress
```

Parameters

Parameters	Description
ByVal lpPrinterAddress As StringBuilder	[out] String buffer which receives the printer address.
ByVal iPrinterAddressBufferSize As Integer	Size of the buffer.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect (see page 120) previously.

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function BSReadPrinterAddress( _
ByVal lpPrinterAddress As StringBuilder _
, ByVal iPrinterAddressBufferSize As Integer _
) As BSError
End Function
```

1.3.1.1.3.19 BodystatAPI.BSReadProtocolInfo Method

Read protocol information from the connected device. Contains version numbers relating to communication protocols in use, data structures and auxiliary information.

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
BSError BSReadProtocolInfo(ByRef Byte pProtocolVersion, ByRef Byte pDataVersion, ByRef Byte
pAuxInfo);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSReadProtocolInfo(ByRef Byte pProtocolVersion, ByRef Byte
pDataVersion, ByRef Byte pAuxInfo);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function BSReadProtocolInfo(ByRef pProtocolVersion As Byte, ByRef
pDataVersion As Byte, ByRef pAuxInfo As Byte) As BSError
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSReadProtocolInfo(ByRef Byte pProtocolVersion, ByRef Byte
pDataVersion, ByRef Byte pAuxInfo);
```

MATLAB

```
BSReadProtocolInfo
```

Parameters

Parameters	Description
ByRef pProtocolVersion As Byte	[out] Receives the communication protocol version.
ByRef pDataVersion As Byte	[out] Receives the data structure version.
ByRef pAuxInfo As Byte	[out] Receives auxiliary information.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

Generally not required for the majority of SDK users as these details are abstracted by the SDK.

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function BSReadProtocolInfo( _
ByRef pProtocolVersion As Byte _
, ByRef pDataVersion As Byte _
, ByRef pAuxInfo As Byte _
) As BSError
End Function
```

1.3.1.1.3.20 BodystatAPI.BSReadSerialNumber Method

Read the unique serial number of the connected Bodystat device.

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
BSError BSReadSerialNumber(ByRef Integer pSerialNumber);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSReadSerialNumber(ByRef Integer pSerialNumber);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function BSReadSerialNumber(ByRef pSerialNumber As Integer) As BSError
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSReadSerialNumber(ByRef Integer pSerialNumber);
```

MATLAB

```
BSReadSerialNumber
```

Parameters

Parameters	Description
ByRef pSerialNumber As Integer	[out] Receives the serial number of the device.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect (see page 120) previously.

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function BSReadSerialNumber( _
ByRef pSerialNumber As Integer _
) As BSError
End Function
```

1.3.1.1.3.21 BodystatAPI.BSReadStoredTestData Method

Download test data from the connected device. Retrieves the last 100 test measurements (records) stored in the memory of the device. Downloading data does not affect the results stored in the memory - they remain unchanged.

Note: you should store downloaded data locally if it is required permanently. Most devices store only the last 100 test records before overwriting the oldest records automatically.

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
BSError BSReadStoredTestData(ByRef BSRawData pRawData);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSReadStoredTestData(ByRef BSRawData pRawData);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function BSReadStoredTestData(ByRef pRawData As BSRawData) As BSError
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSReadStoredTestData(ByRef BSRawData pRawData);
```

MATLAB

```
BSReadStoredTestData
```

Parameters

Parameters	Description
ByRef pRawData As BSRawData	[out] Pointer to a structure to receive the downloaded data. Caller responsible for allocation and disposing of the structure.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect (see page 120) previously. Caller must check they are working with a Bodystat (rather than Equistat model, Equistat models will respond without error - but with garbage results).

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function BSReadStoredTestData( _
ByRef pRawData As BSRawData _
) As BSError
End Function
```

1.3.1.1.3.22 BodystatAPI.BSReadTestBodystat Method

Perform a quick bio-impedance test measurement on the connected bodystat device. Normal test process is skipped (user is not prompted to enter any subject information). Normal test protocol is disregarded (measurement proceeds regardless of lead status). Test measurement is not stored in device memory.

May give wildly erratic readings if leads are not connected properly. Caller should sanity check results to detect for such scenarios. Results are intentionally not sanity checked to provide a true raw reading and provide caller with total discretion.

Useful for diagnostics or performing measurements instigated by software.

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
BSError BSReadTestBodystat(ByRef Integer pZ5, ByRef Integer pZ50, ByRef Integer pZ100,
ByRef Integer pZ200, ByRef Integer pR50, ByRef Single pX50, ByRef Single pPA50);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSReadTestBodystat(ByRef Integer pZ5, ByRef Integer pZ50, ByRef
Integer pZ100, ByRef Integer pZ200, ByRef Integer pR50, ByRef Single pX50, ByRef Single
pPA50);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function BSReadTestBodystat(ByRef pZ5 As Integer, ByRef pZ50 As Integer,
ByRef pZ100 As Integer, ByRef pZ200 As Integer, ByRef pR50 As Integer, ByRef pX50 As
Single, ByRef pPA50 As Single) As BSError
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSReadTestBodystat(ByRef Integer pZ5, ByRef Integer pZ50, ByRef
Integer pZ100, ByRef Integer pZ200, ByRef Integer pR50, ByRef Single pX50, ByRef Single
pPA50);
```

MATLAB

```
BSReadTestBodystat
```

Parameters

Parameters	Description
ByRef pZ5 As Integer	[out] Receives the impedance measured at 5 kHz (result in ohms).
ByRef pZ50 As Integer	[out] Receives the impedance measured at 50 kHz (result in ohms).
ByRef pZ100 As Integer	[out] Receives the impedance measured at 100 kHz (result in ohms).
ByRef pZ200 As Integer	[out] Receives the impedance measured at 200 kHz (result in ohms).
ByRef pR50 As Integer	[out] Receives the resistance measured at 50 kHz (result in ohms).
ByRef pX50 As Single	[out] Receives the reactance measured at 50 kHz (result in ohms).
ByRef pPA50 As Single	[out] Receives the phase angle measured at 50 kHz (result in degrees).

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect (see page 120) previously. Caller must check they are working with a Bodystat (rather than Equistat model, Equistat models will respond without error - but with garbage results).

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function BSReadTestBodystat( _
ByRef pZ5 As Integer _
, ByRef pZ50 As Integer _
, ByRef pZ100 As Integer _
, ByRef pZ200 As Integer _
, ByRef pR50 As Integer _
, ByRef pX50 As Single _
, ByRef pPA50 As Single _
) As BSError
End Function
```

1.3.1.1.3.23 BodystatAPI.BSSearchBTDevices Method

Search BT radio for any Bodystat devices (paired or otherwise). Existing paired devices will be counted even if they are switched off or out of range. New non-paired units will only be counted if switched on, in range and ready for communication (on main startup screen).

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
Integer BSSearchBTDevices(ByRef Integer iNewDevices, ByRef Integer iAuthenticatedDevices,
ByVal Boolean bReportErrors, ByVal IntPtr hWndParent, Optional ByVal UInt16 iTimeout = 7 _);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared Integer BSSearchBTDevices(ByRef Integer iNewDevices, ByRef Integer
iAuthenticatedDevices, ByVal Boolean bReportErrors, ByVal IntPtr hWndParent, Optional ByVal
UInt16 iTimeout);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function BSSearchBTDevices(ByRef iNewDevices As Integer, ByRef
iAuthenticatedDevices As Integer, ByVal bReportErrors As Boolean, ByVal hWndParent As
IntPtr, Optional ByVal iTimeout As UInt16 = 7 _) As Integer
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared Integer BSSearchBTDevices(ByRef Integer iNewDevices, ByRef Integer
iAuthenticatedDevices, ByVal Boolean bReportErrors, ByVal IntPtr hWndParent, Optional ByVal
UInt16 iTimeout);
```

MATLAB

```
BSSearchBTDevices
```

Parameters

Parameters	Description
ByRef iNewDevices As Integer	[out] The number of new Bodystat devices found (not paired).
ByRef iAuthenticatedDevices As Integer	[out] The number of authenticated Bodystat devices (paired).
ByVal bReportErrors As Boolean	TRUE to report errors to the user (shown as message boxes, locale constraints apply). FALSE to hide errors from the user (completely silent).
ByVal hWndParent As IntPtr	Handle of the parent window.
Optional ByVal iTimeout As UInt16 = 7 _	Timeout multiplier supplied to the Bluetooth radio search API. Controls the amount of time to spend searching for the device. This function will block for this duration so is better called from a thread. Each multiplier increment equates to about 1.28 seconds. Thus a timeout value of 10 = 12.8s. Maximum time permitted by the API is a multiplier of 45 (just under 1 minute), values higher than this will be capped at 45. Default value is 7 (about 10 seconds).

Returns

TRUE if it succeeds, FALSE if it fails.

Remarks

Note this process can be time consuming to complete and executes synchronously. Either call from a threaded function or illustrate operation in progress to the user e.g. by means of an indefinite progress bar or busy mouse cursor, until the task completes or timesout.

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function BSSearchBTDevices( _
ByRef iNewDevices As Integer _
, ByRef iAuthenticatedDevices As Integer _
, ByVal bReportErrors As Boolean _
, ByVal hWndParent As IntPtr _
, Optional ByVal iTimeout As UInt16 = 7 _
) As Integer
End Function
```

1.3.1.1.3.24 BodystatAPI.BSUnAuthenticateBTDevices Method

Unauthenticate (unpair) ALL Bodystat devices from this PC.

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
Integer BSUnAuthenticateBTDevices(ByVal Boolean bReportErrors, ByVal IntPtr hWndParent,
Optional ByVal UInt16 iTimeout = 7 _);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared Integer BSUnAuthenticateBTDevices(ByVal Boolean bReportErrors, ByVal IntPtr
hWndParent, Optional ByVal UInt16 iTimeout);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function BSUnAuthenticateBTDevices(ByVal bReportErrors As Boolean, ByVal
hWndParent As IntPtr, Optional ByVal iTimeout As UInt16 = 7 _) As Integer
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared Integer BSUnAuthenticateBTDevices(ByVal Boolean bReportErrors, ByVal IntPtr
hWndParent, Optional ByVal UInt16 iTimeout);
```

MATLAB

```
BSUnAuthenticateBTDevices
```

Parameters

Parameters	Description
ByVal bReportErrors As Boolean	TRUE to report errors to the user (shown as message boxes, locale constraints apply). FALSE to hide errors from the user.
ByVal hWndParent As IntPtr	Handle of the parent window.
Optional ByVal iTimeout As UInt16 = 7 _	Timeout multiplier supplied to the Bluetooth radio search API. Controls the amount of time to spend searching for the device. This function will block for this duration so is better called from a thread. Each multiplier increment equates to about 1.28 seconds. Thus a timeout value of 10 = 12.8s. Maximum time permitted by the API is a multiplier of 45 (just under 1 minute), values higher than this will be capped at 45. Default value is 7 (about 10 seconds).

Returns

TRUE if it succeeds, FALSE if it fails.

Remarks

Note this process can be time consuming to complete and executes synchronously. Either call from a threaded function or illustrate operation in progress to the user e.g. by means of an indefinite progress bar or busy mouse cursor, until the task completes or timesout.

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function BSUnAuthenticateBTDevices( _
ByVal bReportErrors As Boolean _
, ByVal hWndParent As IntPtr _
, Optional ByVal iTimeout As UInt16 = 7 _
) As Integer
End Function
```

1.3.1.1.3.25 BodystatAPI.BSWriteCurrentTime Method

Reset the current date/time of the internal clock on the connected device. Date/time is automatically set to match the current date/time of the PC

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
BSError BSWriteCurrentTime(_);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSWriteCurrentTime(_);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function BSWriteCurrentTime( As _) As BSError
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSWriteCurrentTime(_ );
```

MATLAB

```
BSWriteCurrentTime
```

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect (see page 120) previously. Locale Daylight Saving Time (DST) rules are observed. Use sister function BSWriteCurrentTime(__time64_t CurrentTime) to set a specific date/time which differs from the PC clock or override DST behaviour.

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function BSWriteCurrentTime( _
) As BSError
End Function
```

1.3.1.1.3.26 BSWriteCurrentTimeAs Method

1.3.1.1.3.26.1 BodystatAPI.BSWriteCurrentTimeAs Method (DateTime)

Reset the current date/time of the internal clock on the connected device. Date/time is set to a specific value of your choosing.

C++

```
BSError BSWriteCurrentTimeAs(ByVal DateTime dtCurrentTime);
```

C#

```
public Shared BSError BSWriteCurrentTimeAs(ByVal DateTime dtCurrentTime);
```

Visual Basic

```
Public Shared Function BSWriteCurrentTimeAs(ByVal dtCurrentTime As DateTime) As BSError
```

Java

```
public Shared BSError BSWriteCurrentTimeAs(ByVal DateTime dtCurrentTime);
```

MATLAB

```
BSWriteCurrentTimeAs
```

Parameters

Parameters	Description
ByVal dtCurrentTime As DateTime	Date/time to apply to the internal clock of the device (should be current!).

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect (see page 120) previously. Use sister function (omit date/time parameter entirely) BSWriteCurrentTime (see page 138)() to automatically set to the current date/time of the PC.

Body Source

```
Public Shared Function BSWriteCurrentTimeAs( _
    ByVal dtCurrentTime As DateTime _
    ) As BSError

    Dim dtBaseTime As DateTime = "#1/1/1970#"
    Dim elapsed = dtCurrentTime - dtBaseTime
    Dim llCurrentTime As Int64
    llCurrentTime = elapsed.TotalSeconds

    Return BSWriteCurrentTimeAs(llCurrentTime)
End Function
```

1.3.1.1.3.26.2 BodystatAPI.BSWriteCurrentTimeAs Method (Int64)

Reset the current date/time of the internal clock on the connected device. Date/time is set to a specific value of your choosing.

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
BSError BSWriteCurrentTimeAs(ByVal Int64 CurrentTime);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSWriteCurrentTimeAs(ByVal Int64 CurrentTime);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function BSWriteCurrentTimeAs(ByVal CurrentTime As Int64) As BSError
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSWriteCurrentTimeAs(ByVal Int64 CurrentTime);
```

MATLAB

```
BSWriteCurrentTimeAs
```

Parameters

Parameters	Description
ByVal CurrentTime As Int64	Date/time to apply to the internal clock of the device (should be current!).

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect (see page 120) previously. Use sister function (omit date/time parameter entirely) BSWriteCurrentTime (see page 138)() to automatically set to the current date/time of the PC.

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function BSWriteCurrentTimeAs( _
ByVal CurrentTime As Int64 _
) As BSError
End Function
```

1.3.1.1.3.27 BodystatAPI.BSWritePrinterAddress Method

Set the address of the printer the connected device should use for direct printing. Programmatic implementation of the inbuilt printer pairing function in the device.

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
BSError BSWritePrinterAddress(ByVal StringBuilder lpPrinterAddress, ByVal Integer
iPrinterAddressBufferSize);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSWritePrinterAddress(ByVal StringBuilder lpPrinterAddress, ByVal
Integer iPrinterAddressBufferSize);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function BSWritePrinterAddress(ByVal lpPrinterAddress As StringBuilder, ByVal
iPrinterAddressBufferSize As Integer) As BSError
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError BSWritePrinterAddress(ByVal StringBuilder lpPrinterAddress, ByVal
Integer iPrinterAddressBufferSize);
```

MATLAB

```
BSWritePrinterAddress
```

Parameters

Parameters	Description
ByVal lpPrinterAddress As StringBuilder	String buffer containing the Bluetooth address (BDA) of the printer.
iBufferSize	Size of the buffer.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

Note not all device models support this function (not all support direct printing).

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function BSWritePrinterAddress( _
ByVal lpPrinterAddress As StringBuilder _
, ByVal iPrinterAddressBufferSize As Integer _
) As BSError
End Function
```

1.3.1.1.3.28 BodystatAPI.ESCalculateNormals Method

Calculate normal values and normal ranges for a given animal and their bio-impedance measurement together with the matching results structure for that measurement. You must supply a populated results structure which pertains to the measurement being supplied (obtained from ESCalculateResults (see page 142)).

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
BSError ESCalculateNormals(ByRef ESMeasurement pM, ByRef ESResults pR, ByRef ESNormals pN);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError ESCalculateNormals(ByRef ESMeasurement pM, ByRef ESResults pR, ByRef
ESNormals pN);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function ESCalculateNormals(ByRef pM As ESMeasurement, ByRef pR As ESResults,
ByRef pN As ESNormals) As BSError
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError ESCalculateNormals(ByRef ESMeasurement pM, ByRef ESResults pR, ByRef
ESNormals pN);
```

MATLAB

```
ESCalculateNormals
```

Parameters

Parameters	Description
ByRef pM As ESMeasurement	The measurement details downloaded from the device (includes model, user inputted parameters and bio-impedance measurement readings).
ByRef pR As ESResults	The results calculated for this measurement previously.
ByRef pN As ESNormals	[out] Receives the normal values and normal ranges for this measurement.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

Normals vary by input attributes, measured bio-impedance values and calculated results.

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function ESCalculateNormals( _
ByRef pM As ESMeasurement _
, ByRef pR As ESResults _
, ByRef pN As ESNormals _
) As BSError
End Function
```

1.3.1.1.3.29 BodystatAPI.ESCalculateResults Method

Calculate complete result set for a given bio-impedance measurement recorded by the Equistat device.

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
BSError ESCalculateResults(ByRef ESMeasurement pM, ByRef ESResults pR);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError ESCalculateResults(ByRef ESMeasurement pM, ByRef ESResults pR);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function ESCalculateResults(ByRef pM As ESMeasurement, ByRef pR As ESResults)
As BSError
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError ESCalculateResults(ByRef ESMeasurement pM, ByRef ESResults pR);
```

MATLAB

```
ESCalculateResults
```

Parameters

Parameters	Description
ByRef pM As ESMeasurement	The measurement details downloaded from the device (includes model, user inputted parameters and bio-impedance measurement readings).
ByRef pR As ESResults	[out] Receives the complete calculated results for this measurement.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

Results vary by input attributes and measured bio-impedance values.

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function ESCalculateResults( _
ByRef pM As ESMeasurement _
, ByRef pR As ESResults _
) As BSError
End Function
```

1.3.1.1.3.30 BodystatAPI.ESReadStoredTestData Method

Download test data from the connected device. Retrieves the last 100 test measurements (records) stored in the memory of the device. Downloading data does not affect the results stored in the memory - they remain unchanged.

Note: you should store downloaded data locally if it is required permanently. Most devices store only the last 100 test records before overwriting the oldest records automatically.

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
BSError ESReadStoredTestData(ByRef ESRawData pRawData);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError ESReadStoredTestData(ByRef ESRawData pRawData);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function ESReadStoredTestData(ByRef pRawData As ESRawData) As BSError
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError ESReadStoredTestData(ByRef ESRawData pRawData);
```

MATLAB

```
ESReadStoredTestData
```

Parameters

Parameters	Description
ByRef pRawData As ESRawData	[out] Pointer to a structure to receive the downloaded data. Caller responsible for allocation and disposing of the structure.

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect (see page 120) previously. Caller must check they are working with an Equistat (rather than Bodystat model, Bodystat models will respond without error - but with garbage results).

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function ESReadStoredTestData( _
ByRef pRawData As ESRawData _
) As BSError
End Function
```

1.3.1.1.3.31 BodystatAPI.ESReadTestEquistat Method

Perform a quick bio-impedance test measurement on the connected Equistat device. Normal test process is skipped (user is not prompted to enter any subject information). Normal test protocol is disregarded (measurement proceeds regardless of lead status). Test measurement is not stored in device memory.

May give wildly erratic readings if leads are not connected properly. Caller should sanity check results to detect for such scenarios. Results are intentionally not sanity checked to provide a true raw reading and provide caller with total discretion.

Useful for diagnostics or performing measurements instigated by software.

C++

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
BSError ESReadTestEquistat(ByRef Integer pZ5, ByRef Integer pZ16, ByRef Integer pZ24, ByRef
Integer pZ50, ByRef Integer pZ140, ByRef Integer pZ200, ByRef Integer pZ280, ByRef Single
pPA50);
```

C#

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError ESReadTestEquistat(ByRef Integer pZ5, ByRef Integer pZ16, ByRef
Integer pZ24, ByRef Integer pZ50, ByRef Integer pZ140, ByRef Integer pZ200, ByRef Integer
pZ280, ByRef Single pPA50);
```

Visual Basic

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)>
Public Shared Function ESReadTestEquistat(ByRef pZ5 As Integer, ByRef pZ16 As Integer,
ByRef pZ24 As Integer, ByRef pZ50 As Integer, ByRef pZ140 As Integer, ByRef pZ200 As
Integer, ByRef pZ280 As Integer, ByRef pPA50 As Single) As BSError
```

Java

```
[DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)]
public Shared BSError ESReadTestEquistat(ByRef Integer pZ5, ByRef Integer pZ16, ByRef
Integer pZ24, ByRef Integer pZ50, ByRef Integer pZ140, ByRef Integer pZ200, ByRef Integer
pZ280, ByRef Single pPA50);
```

MATLAB

```
ESReadTestEquistat
```

Parameters

Parameters	Description
ByRef pZ5 As Integer	[out] Receives the impedance measured at 5 kHz (result in ohms).
ByRef pZ16 As Integer	[out] Receives the impedance measured at 16 kHz (result in ohms).
ByRef pZ24 As Integer	[out] Receives the impedance measured at 24 kHz (result in ohms).
ByRef pZ50 As Integer	[out] Receives the impedance measured at 50 kHz (result in ohms).
ByRef pZ140 As Integer	[out] Receives the impedance measured at 140 kHz (result in ohms).
ByRef pZ200 As Integer	[out] Receives the impedance measured at 200 kHz (result in ohms).
ByRef pZ280 As Integer	[out] Receives the impedance measured at 280 kHz (result in ohms).
ByRef pPA50 As Single	[out] Receives the phase angle measured at 50 kHz (result in degress) (ES Pro Only).

Returns

Returns BSError::NoError if successful. Specific error code otherwise.

Remarks

You must have connected to the device by calling BSConnect (see page 120) previously. Caller must check they are working with an Equistat (rather than Bodystat model, Bodystat models will respond without error - but with garbage results).

Body Source

```
<DllImport("BodystatSDK.dll", CallingConvention:=CallingConvention.Cdecl,
CharSet:=CharSet.Unicode)> _
Public Shared Function ESReadTestEquistat( _
ByRef pZ5 As Integer _
, ByRef pZ16 As Integer _
, ByRef pZ24 As Integer _
, ByRef pZ50 As Integer _
, ByRef pZ140 As Integer _
, ByRef pZ200 As Integer _
, ByRef pZ280 As Integer _
, ByRef pPA50 As Single _
) As BSError
End Function
```

1.3.2 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description

	BSAnalysisMode (see page 145)	BSAnalysisMode: Analysis results are available in several formats:
		BSAnalysisModeBC= Body composition analysis
		BSAnalysisModeHN= Hydration analysis
		BSAnalysisModeBoth= Combined body composition and hydration analysis (default)

	BSDeviceFamily (see page 146)	BSDeviceFamily: Enumerated values that group each device / model variant into a given product family. Devices within a given product family share broad capabilities and features. For details regarding capabilities and features refer to the technical documentation for that product family (http://www.bodystat.com)
	BSDeviceModel (see page 147)	BSDeviceModel: Enumerated values that represent the various different models of the Bodystat devices over the years. Suffix: The suffix after the model indicates specific hardware variations regarding connectivity: OPTO indicates model communicates via opto-isolated serial. DIU indicates model communicates via a data interface unit (induction interface) BT indicates model communicates via Bluetooth serial port.
	BSError (see page 149)	BSError: Bodystat Error codes. A list of Bodystat specific error codes.
	BSGender (see page 151)	BSGender: Enumerated values that represent a person's gender.

Module

Visual Basic .NET wrapper (see page 80)

1.3.2.1 BodystatAPI.BSAnalysisMode Enumeration

BSAnalysisMode: Analysis results are available in several formats:

BSAnalysisModeBC= Body composition analysis

BSAnalysisModeHN= Hydration analysis

BSAnalysisModeBoth= Combined body composition and hydration analysis (default)

C++

```
enum BSAnalysisMode {
  BSAnalysisModeNone = 0,
  BSAnalysisModeBC = 1,
  BSAnalysisModeHN = 2,
  BSAnalysisModeBoth = 3
};
```

C#

```
public enum BSAnalysisMode {
  BSAnalysisModeNone = 0,
  BSAnalysisModeBC = 1,
  BSAnalysisModeHN = 2,
  BSAnalysisModeBoth = 3
}
```

Visual Basic

```
Public Enum BSAnalysisMode
  BSAnalysisModeNone = 0
  BSAnalysisModeBC = 1
  BSAnalysisModeHN = 2
  BSAnalysisModeBoth = 3
End Enum
```

Java

```
public class BSAnalysisMode;
```

MATLAB

```
BSAnalysisMode
```

File

BodystatSDK.vb

Members

Members	Description
BSAnalysisModeNone = 0	No analysis mode specified (raw data only)
BSAnalysisModeBC = 1	Body composition enabled
BSAnalysisModeHN = 2	Hydration enabled
BSAnalysisModeBoth = 3	Both body composition and hydration (default)

1.3.2.2 BodystatAPI.BSDeviceFamily Enumeration

BSDeviceFamily: Enumerated values that group each device / model variant into a given product family. Devices within a given product family share broad capabilities and features. For details regarding capabilities and features refer to the technical documentation for that product family (http://www.bodystat.com)

C++

```
enum BSDeviceFamily {
  BSUnknown_Family = 0,
  BS1500_Family = -1,
  BSMDD_Family = -2,
  BSDualScan_Family = -3,
  BSQuadScan_Family = -4,
  BSMultiScan_Family = -5,
  BSEquistatLite_Family = -6,
  BSEquistatPro_Family = -7
};
```

C#

```
public enum BSDeviceFamily {
  BSUnknown_Family = 0,
  BS1500_Family = -1,
  BSMDD_Family = -2,
  BSDualScan_Family = -3,
  BSQuadScan_Family = -4,
  BSMultiScan_Family = -5,
  BSEquistatLite_Family = -6,
  BSEquistatPro_Family = -7
}
```

Visual Basic

```
Public Enum BSDeviceFamily
  BSUnknown_Family = 0
  BS1500_Family = -1
  BSMDD_Family = -2
  BSDualScan_Family = -3
  BSQuadScan_Family = -4
  BSMultiScan_Family = -5
```

```
    BSEquistatLite_Family = -6
    BSEquistatPro_Family = -7
End Enum
```

Java

```
public class BSDeviceFamily;
```

MATLAB

```
BSDeviceFamily
```

File

BodystatSDK.vb

Members

Members	Description
BSUnknown_Family = 0	Unknown product family
BS1500_Family = -1	Bodystat 1500 family
BSMDD_Family = -2	Bodystat MDD family
BSDualScan_Family = -3	Bodystat DualScan family (legacy)
BSQuadScan_Family = -4	Bodystat QuadScan family
BSMultiScan_Family = -5	Bodystat Multiscan family
BSEquistatLite_Family = -6	Equistat Lite family (not supported by this SDK)
BSEquistatPro_Family = -7	Equistat Pro family (not supported by this SDK)

Remarks

Note that not all families are supported by this SDK. Some are listed for historic reference only.

1.3.2.3 BodystatAPI.BSDeviceModel Enumeration

BSDeviceModel: Enumerated values that represent the various different models of the Bodystat devices over the years.

Suffix: The suffix after the model indicates specific hardware variations regarding connectivity:

OPTO indicates model communicates via opto-isolated serial.

DIU indicates model communicates via a data interface unit (induction interface)

BT indicates model communicates via Bluetooth serial port.

C++

```
enum BSDeviceModel {
  BSUnknown = 0,
  BS1500_OPTO = 1,
  BSMDD_OPTO = 2,
  BS1500_DIU = 3,
  BSMDD_DIU = 4,
  BSDualScan_OPTO = 5,
  BSDualScan_DIU = 6,
  BSQuadScan_DIU = 7,
  BSMultiScan_DIU = 8,
  BS1500_BT = 9,
  BSMDD_BT = 10,
  BSQuadScan_BT = 11,
  BSEquistatPro_BT = 12,
  BSEquistatLite_BT = 13,
  BSQuadScanTouch = 14,
  BSMultiScanTouch = 15
};
```

C#

```
public enum BSDeviceModel {
  BSUnknown = 0,
  BS1500_OPTO = 1,
  BSMDD_OPTO = 2,
  BS1500_DIU = 3,
  BSMDD_DIU = 4,
  BSDualScan_OPTO = 5,
  BSDualScan_DIU = 6,
  BSQuadScan_DIU = 7,
  BSMultiScan_DIU = 8,
  BS1500_BT = 9,
  BSMDD_BT = 10,
  BSQuadScan_BT = 11,
  BSEquistatPro_BT = 12,
  BSEquistatLite_BT = 13,
  BSQuadScanTouch = 14,
  BSMultiScanTouch = 15
}
```

Visual Basic

```
  Public Enum BSDeviceModel
    BSUnknown = 0
    BS1500_OPTO = 1
    BSMDD_OPTO = 2
    BS1500_DIU = 3
    BSMDD_DIU = 4
    BSDualScan_OPTO = 5
    BSDualScan_DIU = 6
    BSQuadScan_DIU = 7
    BSMultiScan_DIU = 8
    BS1500_BT = 9
    BSMDD_BT = 10
    BSQuadScan_BT = 11
    BSEquistatPro_BT = 12
    BSEquistatLite_BT = 13
    BSQuadScanTouch = 14
    BSMultiScanTouch = 15
  End Enum
```

Java

```
  public class BSDeviceModel;
```

MATLAB

```
  BSDeviceModel
```

File

BodystatSDK.vb

Members

Members	Description
BSUnknown = 0	Unknown model
BS1500_OPTO = 1	Bodystat 1500 with opto-isolated interface (legacy)
BSMDD_OPTO = 2	Bodystat MDD with opto-isolated interface (legacy)
BS1500_DIU = 3	Bodystat 1500 with data interface unit (legacy)
BSMDD_DIU = 4	Bodystat MDD with data interface unit (legacy)
BSDualScan_OPTO = 5	Bodystat DualScan with opto-isolated interface (legacy)
BSDualScan_DIU = 6	Bodystat DualScan with data interface unit (legacy)
BSQuadScan_DIU = 7	Bodystat QuadScan with data interface unit (legacy)
BSMultiScan_DIU = 8	Bodystat MultiScan with data interface unit (legacy)
BS1500_BT = 9	Bodystat 1500 with Bluetooth interface
BSMDD_BT = 10	Bodystat 1500MDD with Bluetooth interface
BSQuadScan_BT = 11	Bodystat QuadScan 4000 with Bluetooth interface
BSEquistatPro_BT = 12	Equistat Pro with Bluetooth interface (not supported by this SDK)

BSEquistatLite_BT = 13	Equistat Lite with Bluetooth interface (not supported by this SDK)
BSQuadScanTouch = 14	Bodystat QuadScan 4000 Touch with WiFi/Bluetooth interface
BSMultiScanTouch = 15	Bodystat Multiscan with WiFi/Bluetooth interface

Remarks

Note that not all models are supported by this SDK. Some are listed for historic reference only.

1.3.2.4 BodystatAPI.BSError Enumeration

BSError: Bodystat Error codes. A list of Bodystat specific error codes.

C++

```
enum BSError {
  NoError = 0,
  InvalidDeviceModel = -1,
  InvalidDeviceFamily = -2,
  InvalidDeviceMode = -3,
  InvalidGender = -4,
  InvalidAge = -5,
  InvalidHeight = -6,
  InvalidWeight = -7,
  InvalidActivity = -8,
  InvalidWaist = -9,
  InvalidHip = -10,
  InvalidImpedanceZ5 = -11,
  InvalidImpedanceZ50 = -12,
  InvalidImpedanceZ100 = -13,
  InvalidImpedanceZ200 = -14,
  InvalidResistanceR50 = -15,
  InvalidReactanceX50 = -16,
  InvalidPhaseAnglePA50 = -17,
  InvalidParamSupplied = -18,
  ComErrorOpeningPort = -50,
  ComErrorConfiguringPort = -51,
  ComErrorConnecting = -52,
  ComErrorUnsupportedCommand = -53,
  ComErrorProcessingCommand = -54,
  ComErrorReading = -55,
  ComErrorWriting = -56,
  ComErrorUserAborted = -57
};
```

C#

```
public enum BSError {
  NoError = 0,
  InvalidDeviceModel = -1,
  InvalidDeviceFamily = -2,
  InvalidDeviceMode = -3,
  InvalidGender = -4,
  InvalidAge = -5,
  InvalidHeight = -6,
  InvalidWeight = -7,
  InvalidActivity = -8,
  InvalidWaist = -9,
  InvalidHip = -10,
  InvalidImpedanceZ5 = -11,
  InvalidImpedanceZ50 = -12,
  InvalidImpedanceZ100 = -13,
  InvalidImpedanceZ200 = -14,
  InvalidResistanceR50 = -15,
  InvalidReactanceX50 = -16,
  InvalidPhaseAnglePA50 = -17,
  InvalidParamSupplied = -18,
  ComErrorOpeningPort = -50,
  ComErrorConfiguringPort = -51,
  ComErrorConnecting = -52,
  ComErrorUnsupportedCommand = -53,
  ComErrorProcessingCommand = -54,
  ComErrorReading = -55,
  ComErrorWriting = -56,
  ComErrorUserAborted = -57
}
```

Visual Basic

```
Public Enum BSError
  NoError = 0
  InvalidDeviceModel = -1
  InvalidDeviceFamily = -2
  InvalidDeviceMode = -3
  InvalidGender = -4
  InvalidAge = -5
  InvalidHeight = -6
  InvalidWeight = -7
  InvalidActivity = -8
  InvalidWaist = -9
  InvalidHip = -10
  InvalidImpedanceZ5 = -11
  InvalidImpedanceZ50 = -12
  InvalidImpedanceZ100 = -13
  InvalidImpedanceZ200 = -14
  InvalidResistanceR50 = -15
  InvalidReactanceX50 = -16
  InvalidPhaseAnglePA50 = -17
  InvalidParamSupplied = -18
  ComErrorOpeningPort = -50
  ComErrorConfiguringPort = -51
  ComErrorConnecting = -52
  ComErrorUnsupportedCommand = -53
  ComErrorProcessingCommand = -54
  ComErrorReading = -55
  ComErrorWriting = -56
  ComErrorUserAborted = -57
End Enum
```

Java

```
public class BSError;
```

MATLAB

```
BSError
```

File

BodystatSDK.vb

Members

Members	Description
NoError = 0	No error occurred
InvalidDeviceModel = -1	Invalid model specified
InvalidDeviceFamily = -2	Invalid family specified
InvalidDeviceMode = -3	Invalid analysis mode
InvalidGender = -4	Invalid BSGender (see page 151) specified
InvalidAge = -5	Invalid age specified
InvalidHeight = -6	Invalid height measurement specified
InvalidWeight = -7	Invalid weight measurement specified
InvalidActivity = -8	Invalid activity level specified
InvalidWaist = -9	Invalid waist measurement specified
InvalidHip = -10	Invalid hip measurement specified
InvalidImpedanceZ5 = -11	Invalid impedance measurement specified

InvalidImpedanceZ50 = -12	Invalid impedance measurement specified
InvalidImpedanceZ100 = -13	Invalid impedance measurement specified
InvalidImpedanceZ200 = -14	Invalid impedance measurement specified
InvalidResistanceR50 = -15	Invalid resistance measurement specified
InvalidReactanceX50 = -16	Invalid reactance measurement specified
InvalidPhaseAnglePA50 = -17	Invalid phase angle measurement specified
InvalidParamSupplied = -18	Invalid parameter supplied
ComErrorOpeningPort = -50	Error opening the com port
ComErrorConfiguringPort = -51	Error configuring the com port parameters (required baud rate, buffers, stop bits, etc)
ComErrorConnecting = -52	Error connecting to the device
ComErrorUnsupportedCommand = -53	The requested command is not supported by the model being asked to perform it
ComErrorProcessingCommand = -54	Error occurred whilst processing the requested command
ComErrorReading = -55	Error occurred whilst reading data from the com port
ComErrorWriting = -56	Error occurred whilst writing data to the com port
ComErrorUserAborted = -57	The user aborted the process

1.3.2.5 BodystatAPI.BSGender Enumeration

BSGender: Enumerated values that represent a person's gender.

C++

```
enum BSGender {
  BSFemale = 0,
  BSMale = 1
};
```

C#

```
public enum BSGender {
  BSFemale = 0,
  BSMale = 1
}
```

Visual Basic

```
Public Enum BSGender
  BSFemale = 0
  BSMale = 1
End Enum
```

Java

```
public class BSGender;
```

MATLAB

```
BSGender
```

File

BodystatSDK.vb

2 FAQs

Frequently Asked Questions

- C++ Linking Errors:

If you receive errors regarding unresolved symbols when building/linking your project you have most likely forgotten to add the BodystatSDK.lib file to your project.

You can do this by adding a #pragma lib directive to one of your source files like those shown below. An example using this method is also shown in the MFC sample (BodystatMFCDlg.cpp).

```
#if defined _M_X64

#pragma comment ( lib, "..\\..\\BodystatSDK\\Lib\\x64\\BodystatSDK.lib" )

#pragma message ("BodystatMFCDlg.cpp::Linking to 64-bit BodystatSDK.lib")

#else

#pragma comment ( lib, "..\\..\\BodystatSDK\\Lib\\x86\\BodystatSDK.lib" )

#pragma message ("BodystatMFCDlg.cpp::Linking to 32-bit BodystatSDK.lib")

#endif
```

Alternatively, you can reference the Bodystat.lib dependency in the additional dependencies section of your project settings (Project Settings->Configuration Properties->Linker->Input->Additional Dependencies). You will need to specify the file using a relative path to the project or add the path to your VC Library directories.

A simpler alternative still, available to projects in VS2010 (or later), is to simply add the Bodystat.lib file to your solution/project. VS2010 understands .lib files and will do the rest for you.

If you continue to encounter unresolved symbols relating to the Bodystat API, and you have referenced the Bodystat.lib file, another possibility is that your project is using an unsupported character set. The Bodystat SDK is designed to use the Unicode character set. Check your project settings are configured to use Unicode (Project Settings->Configuration Properties->General->Character Set). You may use another character set in your project if you prefer, but ensure your calls to the Bodystat SDK work with Unicode strings.

Once your project builds properly, whichever method you chose, remember you must also copy the BodystatSDK.dll dependency alongside your program.exe or you will see a runtime error. The Bodystat MFC sample automatically copies the dll alongside the app as a pre-build event command line. Ultimately you will need to redistribute the BodystatSDK.DLL in your setup routine for installation along with your application. Remember to reference the appropriate DLL binary for your target platform (e.g. x86). See the Redistribution (see page 154) section for more information.

- .NET Runtime Error: "BadImageFormatException"

If during runtime, your program complains it is unable to load the BodystatSDK.dll because it is an incorrect format you should check the CPU target settings in your configuration. Be careful if you target 'AnyCPU' as this is not presently supported by our SDK (though you can might be able to work around this with careful mappings in your assembly). Instead you simply specify your specific target (x86/x64) then ensure you are referencing the corresponding x86/x64 version of the

BodystatSDK.dll.

3 Redistribution

Redistribution

Once your application is complete and ready for shipping you will need to redistribute the appropriate BodystatSDK.dll together with your application.

The BodystatSDK.DLL can be found in the Lib\x86 or Lib\x64 folder of the Bodystat SDK installation, for 32-bit and 64-bit installations respectively. Simply configure your installation script to install the appropriate DLL alongside your application binaries. Note that 64-bit support is planned and not available at time of writing (please contact us if you have a need for it).

Note: You need only redistribute the BodystatSDK.DLL file. The other .exp and .lib files are only required by your development environment.

The BodystatSDK.dll has no dependencies other that the Windows kernel.

Redistribution of our BodystatSDK.dll remains subject to the terms of our license agreement.

Finally, we recommend that you should always develop and test your software using the latest version of the Bodystat SDK.

4 Samples

Sample projects included within the SDK. The sample projects provide detailed examples which utilise the various functions within the Bodystat API. Samples are provided in a variety of popular programming languages (though we regret we are unable to provide examples for all languages). Please reference the sample in the language you are most familiar with.

Modules

Name	Description
Bodystat MFC Sample (see page 155)	A sample application written in Visual C++ based upon the Microsoft Foundation Classes (MFC). Solutions are provided for Visual Studio 2008 and 2010. You must have Visual C++ 2008 or 2010 installed to work with this sample.
Bodystat C# WinForms Sample (see page 204)	A sample application written in C# .NET based upon WinForms within the Microsoft .NET Framework. Solutions are provided for Visual Studio 2008 and 2010. You must have Visual Studiio 2008 or 2010 installed to work with this sample.
Bodystat Visual Basic .NET Sample (see page 216)	A sample application written in Visual Basic .NET based upon the Microsoft .NET Framework. Solutions are provided for Visual Basic .NET 2008 and 2010. You must have Visual Basic .NET 2008 or 2010 installed to work with this sample.

4.1 Bodystat MFC Sample

A sample application written in Visual C++ based upon the Microsoft Foundation Classes (MFC). Solutions are provided for Visual Studio 2008 and 2010. You must have Visual C++ 2008 or 2010 installed to work with this sample.

Classes

	Name	Description
	CBodystatMFCDlg (see page 156)	CBodystatMFCDlg. Dialog for viewing downloaded Bodystat test measurements records and testing key functions within the Bodystat API.
	CBodystatResultsDlg (see page 185)	CBodystatResultsDlg. Dialog for viewing the Bodystat test measurement results and normals.

4.1.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	CBodystatMFCDlg (see page 156)	CBodystatMFCDlg. Dialog for viewing downloaded Bodystat test measurements records and testing key functions within the Bodystat API.
	CBodystatResultsDlg (see page 185)	CBodystatResultsDlg. Dialog for viewing the Bodystat test measurement results and normals.

Module

Bodystat MFC Sample (see page 155)

4.1.1.1 CBodystatMFCDlg Class

CBodystatMFCDlg. Dialog for viewing downloaded Bodystat test measurements records and testing key functions within the Bodystat API.

Class Hierarchy

CDialogEx
CBodystatMFCDlg

C++

```
class CBodystatMFCDlg : public CDialogEx;
```

C#

```
public class CBodystatMFCDlg : CDialogEx;
```

Visual Basic

```
Public Class CBodystatMFCDlg
Inherits CDialogEx
```

Java

```
public class CBodystatMFCDlg CDialogEx;
```

MATLAB

```
CBodystatMFCDlg
```

File

BodystatMFCDlg.h

Methods

	Name	Description
◆	CBodystatMFCDlg (see page 158)	The main CBodystatMFCDlg constructor

CBodystatMFCDlg Data Members

	Name	Description
◆	m_bsRawData (see page 159)	Used to hold the downloaded test measurements from the device
◆	m_btnAuthenticate (see page 159)	The button authenticate
◆	m_btnAutoSetup (see page 159)	The button automatic setup
◆	m_btnConnect (see page 160)	The button connect
◆	m_btnDisconnect (see page 160)	The button disconnect
◆	m_btnReadCalibrationTime (see page 160)	The button read calibration time
◆	m_btnReadCurrentTime (see page 161)	The button read current time
◆	m_btnReadModelVersion (see page 161)	The button read model version
◆	m_btnReadPrinterAddr (see page 161)	The button read printer address
◆	m_btnReadProtocolInfo (see page 162)	The button read protocol info
◆	m_btnReadSerialNumber (see page 162)	The button read serial number
◆	m_btnReadStoredTestData (see page 162)	The button read stored test data
◆	m_btnReadTestMeasurement (see page 162)	The button read test measurement

	m_btnUnauthenticate (see page 163)	The button unauthenticate
	m_btnWriteCurrentTime (see page 163)	The button write current time
	m_hIcon (see page 163)	Application icon
	m_iTimeout (see page 164)	Bluetooth timeout multiplier. Initial value of 7 (equates to about 10 seconds). This is incremented upon each Bluetooth search. A maximum value of 45 is permitted (about 1 minute).
	m_strComPort (see page 164)	The string com port
	m_TestMeasurementListCtrl (see page 164)	The test measurement list control

CBodystatMFCDlg Methods

	Name	Description
	DoDataExchange (see page 165)	DDX/DDV support Bind the form controls to member variables and set basic input range limits
	OnClickedBtAuthenticateButton (see page 165)	Executes the Bluetooth Authenticate button action. Illustrates use of BSAuthenticateBTDevice (see page 8) from the Bodystat API
	OnClickedBtAutoSetupButton (see page 166)	Executes the Bluetooth Auto Setup button action. Illustrates use of BSAutoSetupBT (see page 8) from the Bodystat API
	OnClickedBtDeviceInfoButton (see page 167)	Executes the Bluetooth Device Info button action. Illustrates use of BSGetBTBodystatDevice (see page 12) from the Bodystat API
	OnClickedBtSearchButton (see page 167)	Executes the Bluetooth Search button action. Illustrates use of BSSearchBTDevices (see page 22) from the Bodystat API
	OnClickedBtStackInfoButton (see page 168)	Executes the Bluetooth Get Stack Info button action. Illustrates use of BSGetBTStackInfo (see page 13) from the Bodystat API
	OnClickedBtStatusButton (see page 169)	Executes the Bluetooth Get Status button action. Illustrates use of BSIsBTAvailable (see page 16) from the Bodystat API
	OnClickedBtUnauthenticateButton (see page 169)	Executes the Bluetooth Unauthenticate button action. Illustrates use of BSUnAuthenticateBTDevices (see page 23) from the Bodystat API
	OnClickedComConnectButton (see page 170)	Executes the communication Connect button action. Illustrates use of BSOpenComport (see page 16) from the Bodystat API
	OnClickedComReadCalibrationTimeButton (see page 170)	Executes the communication Read Calibration Time button action. Illustrates use of BSReadCalibrationTime (see page 17) from the Bodystat API
	OnClickedComReadCurrentTimeButton (see page 171)	Executes the communication Read Current Time button action. Illustrates use of BSReadCurrentTime (see page 17) from the Bodystat API
	OnClickedComReadModelVersionButton (see page 172)	Executes the communication Read Model Version button action. Illustrates use of BSReadModelVersion (see page 18) from the Bodystat API
	OnClickedComReadPrinterAddrButton (see page 173)	Executes the communication Read Printer Address button action. Illustrates use of BSReadPrinterAddress (see page 19) from the Bodystat API
	OnClickedComReadProtocolInfoButton (see page 174)	Executes the communication ReadProtocolInfo button action. Illustrates use of BSReadProtocolInfo (see page 19) from the Bodystat API
	OnClickedComReadSerialNumberButton (see page 175)	Executes the communication Read Serial Number button action. Illustrates use of BSReadSerialNumber (see page 20) from the Bodystat API

	OnClickedComReadStoredTestDataButton (see page 175)	Executes the communication Read Stored Test Data button action. Illustrates use of BSReadStoredTestData (see page 21) from the Bodystat API to download data
	OnClickedComReadTestMeasurementButton (see page 176)	Executes the communication Read Test Measurement button action. Performs an immediate test measurement on the device using the BSReadTestBodystat (see page 21) function from the Bodystat API
	OnClickedComWriteCurrentTimeButton (see page 177)	Executes the communication Write Current Time button action. Illustrates use of BSWriteCurrentTime (see page 24) from the Bodystat API
	OnClickedResultsButton (see page 178)	Executes the Results button action. Displays the results dialog with the selected test measurement or a sample result if none is selected.
	OnDblclkTestMeasurementList (see page 179)	Executes when an item in the measurement list is double clicked.Displays the results dialog with the selected test measurement (or a sample result if none is selected.)
	OnInitDialog (see page 179)	Generated message map functions CBodystatMFCDlg message handlers
	OnPaint (see page 180)	If you add a minimize button to your dialog, you will need the code below to draw the icon. For MFC applications using the document/view model, this is automatically done for you by the framework.
	OnQueryDragIcon (see page 181)	The system calls this function to obtain the cursor to display while the user drags the minimized window.
	OnSysCommand (see page 181)	This is OnSysCommand, a member of class CBodystatMFCDlg.
	PopulateMeasurementList (see page 182)	Populate the measurement list control with test measurement data from the API. Illustrates iteration through a BSRawData (see page 40) structure, working with data in a BSMeasurement (see page 36) structure and helper functions from the Bodystat API like BSGetDeviceModelName (see page 14)
	ShowResultsDialog (see page 184)	Displays the results dialog with the selected test measurement (or a sample result if none is selected.)

4.1.1.1.1 CBodystatMFCDlg::CBodystatMFCDlg Constructor

The main CBodystatMFCDlg constructor

C++

```
CBodystatMFCDlg(CWnd* pParent = NULL);
```

C#

```
public CBodystatMFCDlg(ref CWnd pParent);
```

Visual Basic

```
Public Sub New(ByRef pParent As CWnd = NULL)
```

Java

```
public CBodystatMFCDlg(CWnd pParent);
```

MATLAB

```
CBodystatMFCDlg
```

Parameters

Parameters	Description
CWnd* pParent = NULL	=NULL

Description

standard constructor

Body Source

```
CBodystatMFCDlg::CBodystatMFCDlg(CWnd* pParent /*=NULL*/)
 : CDialogEx(CBodystatMFCDlg::IDD, pParent)
 , m_strComPort(_T(""))
{
 m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
 m_iTimeout = 7; // Initial Bluetooth timeout multiplier of 7 (equates to about 10
seconds). This is incremented upon each Bluetooth search. A maximum value of 45 is
permitted (about 1 minute).


}
```

4.1.1.1.2 CBodystatMFCDlg Data Members

4.1.1.1.2.1 CBodystatMFCDlg::m_bsRawData Data Member

C++

```
BSRawData m_bsRawData;
```

C#

```
protected BSRawData m_bsRawData;
```

Visual Basic

```
Protected m_bsRawData As BSRawData
```

Java

```
protected BSRawData m_bsRawData;
```

MATLAB

```
m_bsRawData
```

Description

Used to hold the downloaded test measurements from the device

4.1.1.1.2.2 CBodystatMFCDlg::m_btnAuthenticate Data Member

The button authenticate

C++

```
CButton m_btnAuthenticate;
```

C#

```
protected CButton m_btnAuthenticate;
```

Visual Basic

```
Protected m_btnAuthenticate As CButton
```

Java

```
protected CButton m_btnAuthenticate;
```

MATLAB

```
m_btnAuthenticate
```

4.1.1.1.2.3 CBodystatMFCDlg::m_btnAutoSetup Data Member

The button automatic setup

C++

```
CButton m_btnAutoSetup;
```

C#

```
protected CButton m_btnAutoSetup;
```

Visual Basic

```
Protected m_btnAutoSetup As CButton
```

Java

```
protected CButton m_btnAutoSetup;
```

MATLAB

```
m_btnAutoSetup
```

4.1.1.1.2.4 CBodystatMFCDlg::m_btnConnect Data Member

The button connect

C++

```
CButton m_btnConnect;
```

C#

```
protected CButton m_btnConnect;
```

Visual Basic

```
Protected m_btnConnect As CButton
```

Java

```
protected CButton m_btnConnect;
```

MATLAB

```
m_btnConnect
```

4.1.1.1.2.5 CBodystatMFCDlg::m_btnDisconnect Data Member

The button disconnect

C++

```
CButton m_btnDisconnect;
```

C#

```
protected CButton m_btnDisconnect;
```

Visual Basic

```
Protected m_btnDisconnect As CButton
```

Java

```
protected CButton m_btnDisconnect;
```

MATLAB

```
m_btnDisconnect
```

4.1.1.1.2.6 CBodystatMFCDlg::m_btnReadCalibrationTime Data Member

The button read calibration time

C++

```
CButton m_btnReadCalibrationTime;
```

C#

```
protected CButton m_btnReadCalibrationTime;
```

Visual Basic

```
Protected m_btnReadCalibrationTime As CButton
```

Java

```
protected CButton m_btnReadCalibrationTime;
```

MATLAB

```
m_btnReadCalibrationTime
```

4.1.1.1.2.7 CBodystatMFCDlg::m_btnReadCurrentTime Data Member

The button read current time

C++

```
CButton m_btnReadCurrentTime;
```

C#

```
protected CButton m_btnReadCurrentTime;
```

Visual Basic

```
Protected m_btnReadCurrentTime As CButton
```

Java

```
protected CButton m_btnReadCurrentTime;
```

MATLAB

```
m_btnReadCurrentTime
```

4.1.1.1.2.8 CBodystatMFCDlg::m_btnReadModelVersion Data Member

The button read model version

C++

```
CButton m_btnReadModelVersion;
```

C#

```
protected CButton m_btnReadModelVersion;
```

Visual Basic

```
Protected m_btnReadModelVersion As CButton
```

Java

```
protected CButton m_btnReadModelVersion;
```

MATLAB

```
m_btnReadModelVersion
```

4.1.1.1.2.9 CBodystatMFCDlg::m_btnReadPrinterAddr Data Member

The button read printer address

C++

```
CButton m_btnReadPrinterAddr;
```

C#

```
protected CButton m_btnReadPrinterAddr;
```

Visual Basic

```
Protected m_btnReadPrinterAddr As CButton
```

Java

```
protected CButton m_btnReadPrinterAddr;
```

MATLAB

```
m_btnReadPrinterAddr
```

4.1.1.1.2.10 CBodystatMFCDlg::m_btnReadProtocolInfo Data Member

The button read protocol info

C++

```
CButton m_btnReadProtocolInfo;
```

C#

```
protected CButton m_btnReadProtocolInfo;
```

Visual Basic

```
Protected m_btnReadProtocolInfo As CButton
```

Java

```
protected CButton m_btnReadProtocolInfo;
```

MATLAB

```
m_btnReadProtocolInfo
```

4.1.1.1.2.11 CBodystatMFCDlg::m_btnReadSerialNumber Data Member

The button read serial number

C++

```
CButton m_btnReadSerialNumber;
```

C#

```
protected CButton m_btnReadSerialNumber;
```

Visual Basic

```
Protected m_btnReadSerialNumber As CButton
```

Java

```
protected CButton m_btnReadSerialNumber;
```

MATLAB

```
m_btnReadSerialNumber
```

4.1.1.1.2.12 CBodystatMFCDlg::m_btnReadStoredTestData Data Member

The button read stored test data

C++

```
CButton m_btnReadStoredTestData;
```

C#

```
protected CButton m_btnReadStoredTestData;
```

Visual Basic

```
Protected m_btnReadStoredTestData As CButton
```

Java

```
protected CButton m_btnReadStoredTestData;
```

MATLAB

```
m_btnReadStoredTestData
```

4.1.1.1.2.13 CBodystatMFCDlg::m_btnReadTestMeasurement Data Member

The button read test measurement

C++

```
CButton m_btnReadTestMeasurement;
```

C#

```
protected CButton m_btnReadTestMeasurement;
```

Visual Basic

```
Protected m_btnReadTestMeasurement As CButton
```

Java

```
protected CButton m_btnReadTestMeasurement;
```

MATLAB

```
m_btnReadTestMeasurement
```

4.1.1.1.2.14 CBodystatMFCDlg::m_btnUnauthenticate Data Member

The button unauthenticate

C++

```
CButton m_btnUnauthenticate;
```

C#

```
protected CButton m_btnUnauthenticate;
```

Visual Basic

```
Protected m_btnUnauthenticate As CButton
```

Java

```
protected CButton m_btnUnauthenticate;
```

MATLAB

```
m_btnUnauthenticate
```

4.1.1.1.2.15 CBodystatMFCDlg::m_btnWriteCurrentTime Data Member

The button write current time

C++

```
CButton m_btnWriteCurrentTime;
```

C#

```
protected CButton m_btnWriteCurrentTime;
```

Visual Basic

```
Protected m_btnWriteCurrentTime As CButton
```

Java

```
protected CButton m_btnWriteCurrentTime;
```

MATLAB

```
m_btnWriteCurrentTime
```

4.1.1.1.2.16 CBodystatMFCDlg::m_hIcon Data Member

C++

```
HICON m_hIcon;
```

C#

```
protected HICON m_hIcon;
```

Visual Basic

```
Protected m_hIcon As HICON
```

Java

```
protected HICON m_hIcon;
```

MATLAB

```
m_hIcon
```

Description

Application icon

4.1.1.1.2.17 CBodystatMFCDlg::m_iTimeout Data Member

C++

```
unsigned short m_iTimeout;
```

C#

```
protected unsigned short m_iTimeout;
```

Visual Basic

```
Protected m_iTimeout As unsigned short
```

Java

```
protected unsigned short m_iTimeout;
```

MATLAB

```
m_iTimeout
```

Description

Bluetooth timeout multiplier. Initial value of 7 (equates to about 10 seconds). This is incremented upon each Bluetooth search. A maximum value of 45 is permitted (about 1 minute).

4.1.1.1.2.18 CBodystatMFCDlg::m_strComPort Data Member

The string com port

C++

```
CString m_strComPort;
```

C#

```
protected CString m_strComPort;
```

Visual Basic

```
Protected m_strComPort As CString
```

Java

```
protected CString m_strComPort;
```

MATLAB

```
m_strComPort
```

4.1.1.1.2.19 CBodystatMFCDlg::m_TestMeasurementListCtrl Data Member

The test measurement list control

C++

```
CListCtrl m_TestMeasurementListCtrl;
```

C#

```
protected CListCtrl m_TestMeasurementListCtrl;
```

Visual Basic

```
Protected m_TestMeasurementListCtrl As CListCtrl
```

Java

```
protected CListCtrl m_TestMeasurementListCtrl;
```

MATLAB

```
m_TestMeasurementListCtrl
```

4.1.1.1.3 CBodystatMFCDlg Methods

4.1.1.1.3.1 CBodystatMFCDlg::DoDataExchange Method

C++

```
virtual void DoDataExchange(CDataExchange* pDX);
```

C#

```
protected DoDataExchange(ref CDataExchange pDX);
```

Visual Basic

```
Protected Sub DoDataExchange(ByRef pDX As CDataExchange)
```

Java

```
protected DoDataExchange(CDataExchange pDX);
```

MATLAB

```
DoDataExchange
```

Description

DDX/DDV support

Bind the form controls to member variables and set basic input range limits

Body Source

```
void CBodystatMFCDlg::DoDataExchange(CDataExchange* pDX)
{
 CDialogEx::DoDataExchange(pDX);

 DDX_Control(pDX, IDC_BT_AUTOSETUP_BUTTON, m_btnAutoSetup);
 DDX_Control(pDX, IDC_BT_AUTHENTICATE_BUTTON, m_btnAuthenticate);
 DDX_Control(pDX, IDC_BT_UNAUTHENTICATE_BUTTON, m_btnUnauthenticate);

 DDX_Text(pDX, IDC_COMPORT_EDIT, m_strComPort);
 DDX_Control(pDX, IDC_COM_CONNECT_BUTTON, m_btnConnect);
 DDX_Control(pDX, IDC_COM_READCALIBRATIONTIME_BUTTON, m_btnReadCalibrationTime);
 DDX_Control(pDX, IDC_COM_READCURRENTTIME_BUTTON, m_btnReadCurrentTime);
 DDX_Control(pDX, IDC_COM_READMODELVERSION_BUTTON, m_btnReadModelVersion);
 DDX_Control(pDX, IDC_COM_READPRINTERADDR_BUTTON, m_btnReadPrinterAddr);
 DDX_Control(pDX, IDC_COM_READPROTOCOLINFO_BUTTON, m_btnReadProtocolInfo);
 DDX_Control(pDX, IDC_COM_READSERIALNUMBER_BUTTON, m_btnReadSerialNumber);
 DDX_Control(pDX, IDC_COM_READSTOREDTESTDATA_BUTTON, m_btnReadStoredTestData);
 DDX_Control(pDX, IDC_COM_READTESTMEASUREMENT_BUTTON, m_btnReadTestMeasurement);
 DDX_Control(pDX, IDC_COM_WRITECURRENTTIME_BUTTON, m_btnWriteCurrentTime);
 DDX_Control(pDX, IDC_TESTMEASUREMENT_LIST, m_TestMeasurementListCtrl);
}
```

4.1.1.1.3.2 CBodystatMFCDlg::OnClickedBtAuthenticateButton Method

Executes the Bluetooth Authenticate button action. Illustrates use of BSAuthenticateBTDevice (see page 8) from the Bodystat API

C++

```
afx_msg void OnClickedBtAuthenticateButton();
```

C#

```
public afx_msg void OnClickedBtAuthenticateButton();
```

Visual Basic

```
Public Function OnClickedBtAuthenticateButton() As afx_msg void
```

Java

```
public afx_msg void OnClickedBtAuthenticateButton();
```

MATLAB

```
OnClickedBtAuthenticateButton
```

Body Source

```
void CBodystatMFCDlg::OnClickedBtAuthenticateButton()
{
 CWaitCursor wc;  // Lengthy operation, show the user we are doing something

 if(BSAuthenticateBTDevice(GetSafeHwnd(), m_iTimeout))
 {
  AfxMessageBox(_T("Device authenticated\nClick Device Info button for more information"),
MB_ICONINFORMATION);
 }
 else
  AfxMessageBox(_T("Error no Bodystat device could be found or authenticated"));
}
```

4.1.1.1.3.3 CBodystatMFCDlg::OnClickedBtAutoSetupButton Method

Executes the Bluetooth Auto Setup button action. Illustrates use of BSAutoSetupBT (see page 8) from the Bodystat API

C++

```
afx_msg void OnClickedBtAutoSetupButton();
```

C#

```
public afx_msg void OnClickedBtAutoSetupButton();
```

Visual Basic

```
Public Function OnClickedBtAutoSetupButton() As afx_msg void
```

Java

```
public afx_msg void OnClickedBtAutoSetupButton();
```

MATLAB

```
OnClickedBtAutoSetupButton
```

Body Source

```
void CBodystatMFCDlg::OnClickedBtAutoSetupButton()
{
 CString strComPort;

 CWaitCursor wc;  // Lengthy operation, show the user we are doing something

 // Perform auto setup. User will be prompted as needed by the API
 // Suitable for supported locales only. Non-supported locales will display messages in
English
 // For non-supported locales you should use the search and authenticate functions directly
(silently, prompting the user yourself as required)
 if(BSAutoSetupBT(strComPort.GetBuffer(256), 256, TRUE, GetSafeHwnd()))
 {
  CString strMsg;
  strMsg.Format(_T("Device setup complete.\nCom Port: %s\nClick Device Info button for more
information"), strComPort);

  UpdateData(TRUE);
  m_strComPort = strComPort;  // Assign this port to the edit box on the form
  UpdateData(FALSE);

  AfxMessageBox(strMsg, MB_ICONINFORMATION);
 }
 else
  AfxMessageBox(_T("Error no Bodystat device could be found or authenticated"));

 strComPort.ReleaseBuffer();
}
```

4.1.1.1.3.4 CBodystatMFCDlg::OnClickedBtDeviceInfoButton Method

Executes the Bluetooth Device Info button action. Illustrates use of BSGetBTBodystatDevice (see page 12) from the Bodystat API

C++

```
afx_msg void OnClickedBtDeviceInfoButton();
```

C#

```
public afx_msg void OnClickedBtDeviceInfoButton();
```

Visual Basic

```
Public Function OnClickedBtDeviceInfoButton() As afx_msg void
```

Java

```
public afx_msg void OnClickedBtDeviceInfoButton();
```

MATLAB

```
OnClickedBtDeviceInfoButton
```

Body Source

```
void CBodystatMFCDlg::OnClickedBtDeviceInfoButton()
{
 CString strDeviceName;
 CString strComPort;
 CString strBDA;

 CWaitCursor wc; // Lengthy operation (if not already searched), show the user we are doing
something

 // Retrieve information about the Bodystat device
 BOOL bSuccess = BSGetBTBodystatDevice(strDeviceName.GetBuffer(512), 512,
strBDA.GetBuffer(18), 18, strComPort.GetBuffer(256), 256, m_iTimeout);
 strDeviceName.ReleaseBuffer();
 strComPort.ReleaseBuffer();
 strBDA.ReleaseBuffer();

 if(bSuccess)
 {
  CString strMsg;
  strMsg.Format(_T("Found Bodystat device.\n\nDevice Name: %s\nBluetooth Address: %s\nCom
Port: %s"), strDeviceName, strBDA, strComPort.IsEmpty() ? _T("<requires authentication>") :
strComPort);

  UpdateData(TRUE);
  m_strComPort = strComPort;  // Assign this port to the edit box on the form
  UpdateData(FALSE);

  AfxMessageBox(strMsg, MB_ICONINFORMATION);
 }
 else
  AfxMessageBox(_T("Error no Bodystat device was found.\nClick Search Devices button to
retry."));
}
```

4.1.1.1.3.5 CBodystatMFCDlg::OnClickedBtSearchButton Method

Executes the Bluetooth Search button action. Illustrates use of BSSearchBTDevices (see page 22) from the Bodystat API

C++

```
afx_msg void OnClickedBtSearchButton();
```

C#

```
public afx_msg void OnClickedBtSearchButton();
```

Visual Basic

```
Public Function OnClickedBtSearchButton() As afx_msg void
```

Java

```
public afx_msg void OnClickedBtSearchButton();
```

MATLAB

```
OnClickedBtSearchButton
```

Body Source

```
void CBodystatMFCDlg::OnClickedBtSearchButton()
{
 int iNewDevices = 0;
 int iAuthenticatedDevices = 0;

 CWaitCursor wc; // Lengthy operation, show the user we are doing something

 // Begin the search...
 if(BSSearchBTDevices(iNewDevices, iAuthenticatedDevices, TRUE, GetSafeHwnd(), m_iTimeout))
 {
  CString strMsg;
  strMsg.Format(_T("Search complete.\nFound %d new and %d previously authenticated Bodystat
Bluetooth devices.\nSearch waited approximately %.2fs."), iNewDevices,
iAuthenticatedDevices, m_iTimeout * 1.28);
  AfxMessageBox(strMsg, MB_ICONINFORMATION);

  // If no new devices were found, lets try extending the search time for next time around
  // We can wait up to the maximum of value of 45 - about 1 minute
  // Obviously lengthy waits like this need to be in a background thread in a real world
example
  if (iNewDevices == 0)
   m_iTimeout = min(m_iTimeout * 2, 45);
 }
 else
  AfxMessageBox(_T("Error searching for Bluetooth devices"));
}
```

4.1.1.1.3.6 CBodystatMFCDlg::OnClickedBtStackInfoButton Method

Executes the Bluetooth Get Stack Info button action. Illustrates use of BSGetBTStackInfo (see page 13) from the Bodystat API

C++

```
afx_msg void OnClickedBtStackInfoButton();
```

C#

```
public afx_msg void OnClickedBtStackInfoButton();
```

Visual Basic

```
Public Function OnClickedBtStackInfoButton() As afx_msg void
```

Java

```
public afx_msg void OnClickedBtStackInfoButton();
```

MATLAB

```
OnClickedBtStackInfoButton
```

Body Source

```
void CBodystatMFCDlg::OnClickedBtStackInfoButton()
{
 CString strInfo;
 if(BSGetBTStackInfo(strInfo.GetBuffer(1024), 1024))
  AfxMessageBox(strInfo, MB_ICONINFORMATION);
 else
  AfxMessageBox(_T("Error: No compatible Bluetooth Stack is available in this pc.\nEither
it is switched off, or it is not recognised by this software.\nOnly Microsoft and Widcomm
stacks are supported by Bodystat's SDK."));
```

```
 strInfo.ReleaseBuffer();
}
```

4.1.1.1.3.7 CBodystatMFCDlg::OnClickedBtStatusButton Method

Executes the Bluetooth Get Status button action. Illustrates use of BSIsBTAvailable (see page 16) from the Bodystat API

C++

```
afx_msg void OnClickedBtStatusButton();
```

C#

```
public afx_msg void OnClickedBtStatusButton();
```

Visual Basic

```
Public Function OnClickedBtStatusButton() As afx_msg void
```

Java

```
public afx_msg void OnClickedBtStatusButton();
```

MATLAB

```
OnClickedBtStatusButton
```

Body Source

```
void CBodystatMFCDlg::OnClickedBtStatusButton()
{
 if(BSIsBTAvailable())
  AfxMessageBox(_T("Supported Bluetooth radio module is available!"), MB_ICONINFORMATION);
 else
  AfxMessageBox(_T("No compatible Bluetooth radio module is available in this pc.\nEither
it is switched off, or it is not recognised by this software.\nOnly Microsoft and Widcomm
stacks are supported by Bodystat's SDK."));
}
```

4.1.1.1.3.8 CBodystatMFCDlg::OnClickedBtUnauthenticateButton Method

Executes the Bluetooth Unauthenticate button action. Illustrates use of BSUnAuthenticateBTDevices (see page 23) from the Bodystat API

C++

```
afx_msg void OnClickedBtUnauthenticateButton();
```

C#

```
public afx_msg void OnClickedBtUnauthenticateButton();
```

Visual Basic

```
Public Function OnClickedBtUnauthenticateButton() As afx_msg void
```

Java

```
public afx_msg void OnClickedBtUnauthenticateButton();
```

MATLAB

```
OnClickedBtUnauthenticateButton
```

Body Source

```
void CBodystatMFCDlg::OnClickedBtUnauthenticateButton()
{
 CWaitCursor wc;  // Lengthy operation, show the user we are doing something

 if(BSUnAuthenticateBTDevices(TRUE, GetSafeHwnd(),m_iTimeout))
 {
  AfxMessageBox(_T("Devices authentication removed"), MB_ICONINFORMATION);
 }
 else
  AfxMessageBox(_T("Error removing authentication"));

}
```

4.1.1.1.3.9 CBodystatMFCDlg::OnClickedComConnectButton Method

Executes the communication Connect button action. Illustrates use of BSOpenComport (see page 16) from the Bodystat API

C++

```
afx_msg void OnClickedComConnectButton();
```

C#

```
public afx_msg void OnClickedComConnectButton();
```

Visual Basic

```
Public Function OnClickedComConnectButton() As afx_msg void
```

Java

```
public afx_msg void OnClickedComConnectButton();
```

MATLAB

```
OnClickedComConnectButton
```

Body Source

```
void CBodystatMFCDlg::OnClickedComConnectButton()
{
 UpdateData(TRUE);

 CWaitCursor wc;  // Lengthy operation, show the user we are doing something

 // Open a com port to the device
 BSError bsErr = BSOpenComport(m_strComPort.GetBuffer(10), 10);
 if(bsErr == Bodystat::NoError)
 {
  bsErr = BSConnect();
  if(bsErr == Bodystat::NoError)
   AfxMessageBox(_T("Connected to device successfully."), MB_ICONINFORMATION);
  else
   AfxMessageBox(_T("Error connecting to device."));

  BSCloseComport();
 }
 else
  AfxMessageBox(_T("Error opening com port."));
}
```

4.1.1.1.3.10 CBodystatMFCDlg::OnClickedComReadCalibrationTimeButton Method

Executes the communication Read Calibration Time button action. Illustrates use of BSReadCalibrationTime (see page 17) from the Bodystat API

C++

```
afx_msg void OnClickedComReadCalibrationTimeButton();
```

C#

```
public afx_msg void OnClickedComReadCalibrationTimeButton();
```

Visual Basic

```
Public Function OnClickedComReadCalibrationTimeButton() As afx_msg void
```

Java

```
public afx_msg void OnClickedComReadCalibrationTimeButton();
```

MATLAB

```
OnClickedComReadCalibrationTimeButton
```

Body Source

```
void CBodystatMFCDlg::OnClickedComReadCalibrationTimeButton()
```

```
{
 UpdateData(TRUE); // Grab the com port value from the edit box of the form

 CWaitCursor wc;  // Lengthy operation, show the user we are doing something

 // Open a com port to the device
 BSError bsErr = BSOpenComport(m_strComPort.GetBuffer(10), 10);
 if(bsErr == Bodystat::NoError)
 {
  // Connect to the device
  bsErr = BSConnect();
  if(bsErr == Bodystat::NoError)
  {
   __time64_t time;

   // Read the calibration date from the device
   bsErr = BSReadCalibrationTime(&time);
   if(bsErr == Bodystat::NoError)
   {
    COleDateTime dtTime(time);
    CString strMsg;
    strMsg.Format(_T("Queried device successfully\nCalibration Date: %s"),
dtTime.Format(VAR_DATEVALUEONLY));  // Only interested in the date part of calibration date
    AfxMessageBox(strMsg, MB_ICONINFORMATION);
   }
   else
    AfxMessageBox(_T("Error querying calibration date/time from device.\nNote that legacy
devices do store their calibration date electronically."));
  }
  else
   AfxMessageBox(_T("Error connecting to device."));

  BSCloseComport();
 }
 else
  AfxMessageBox(_T("Error opening com port."));
}
```

4.1.1.1.3.11 CBodystatMFCDlg::OnClickedComReadCurrentTimeButton Method

Executes the communication Read Current Time button action. Illustrates use of BSReadCurrentTime (see page 17) from the Bodystat API

C++

```
afx_msg void OnClickedComReadCurrentTimeButton();
```

C#

```
public afx_msg void OnClickedComReadCurrentTimeButton();
```

Visual Basic

```
Public Function OnClickedComReadCurrentTimeButton() As afx_msg void
```

Java

```
public afx_msg void OnClickedComReadCurrentTimeButton();
```

MATLAB

```
OnClickedComReadCurrentTimeButton
```

Body Source

```
void CBodystatMFCDlg::OnClickedComReadCurrentTimeButton()
{
 UpdateData(TRUE); // Grab the com port value from the edit box of the form

 CWaitCursor wc;  // Lengthy operation, show the user we are doing something

 // Open a com port to the device
 BSError bsErr = BSOpenComport(m_strComPort.GetBuffer(10), 10);
 if(bsErr == Bodystat::NoError)
 {
  // Connect to the device
  bsErr = BSConnect();
  if(bsErr == Bodystat::NoError)
  {
   __time64_t time;

   // Read the current time from the internal clock of the device
   bsErr = BSReadCurrentTime(&time);
   if(bsErr == Bodystat::NoError)
   {
    COleDateTime dtTime(time);  // Convert it to an MFC COleDateTime for easy display
    CString strMsg;
    strMsg.Format(_T("Queried device successfully\nCurrent Date/Time: %s"),
dtTime.Format());
    AfxMessageBox(strMsg, MB_ICONINFORMATION);
   }
   else
    AfxMessageBox(_T("Error querying current date/time from device.\nNot all models have an
inbuilt real time clock."));
  }
  else
   AfxMessageBox(_T("Error connecting to device."));

  BSCloseComport();
 }
 else
  AfxMessageBox(_T("Error opening com port."));
}
```

4.1.1.1.3.12 CBodystatMFCDlg::OnClickedComReadModelVersionButton Method

Executes the communication Read Model Version button action. Illustrates use of BSReadModelVersion (see page 18) from the Bodystat API

C++

```
afx_msg void OnClickedComReadModelVersionButton();
```

C#

```
public afx_msg void OnClickedComReadModelVersionButton();
```

Visual Basic

```
Public Function OnClickedComReadModelVersionButton() As afx_msg void
```

Java

```
public afx_msg void OnClickedComReadModelVersionButton();
```

MATLAB

```
OnClickedComReadModelVersionButton
```

Body Source

```
void CBodystatMFCDlg::OnClickedComReadModelVersionButton()
{
 UpdateData(TRUE); // Grab the com port value from the edit box of the form

 CWaitCursor wc;  // Lengthy operation, show the user we are doing something

 // Open a com port to the device
 BSError bsErr = BSOpenComport(m_strComPort.GetBuffer(10), 10);
 if(bsErr == Bodystat::NoError)
 {
  // Connect to the device
  bsErr = BSConnect();
  if(bsErr == Bodystat::NoError)
  {
   BSDeviceModel iModel;
   BYTE byMajor;
   BYTE byMinor;
   BYTE byPsoc2Version;
   BYTE byEepromVersion;
```

```
    BYTE byBluetoothInfo;

    // Read Model & Version information from the device
    bsErr = BSReadModelVersion(&iModel, &byMajor, &byMinor, &byPsoc2Version,
&byEepromVersion, &byBluetoothInfo);
    if(bsErr == Bodystat::NoError)
    {
     CString strMsg;
     strMsg.Format(_T("Queried device successfully\n\nDevice Type: %d\nFirmware Version:
%d.%d.%d.%d"), iModel, byMajor, byMinor, byPsoc2Version, byEepromVersion);
     AfxMessageBox(strMsg, MB_ICONINFORMATION);
    }
    else
     AfxMessageBox(_T("Error querying model information from device."));
   }
   else
    AfxMessageBox(_T("Error connecting to device."));

   BSCloseComport();
  }
  else
   AfxMessageBox(_T("Error opening com port."));
 }
```

4.1.1.1.3.13 CBodystatMFCDlg::OnClickedComReadPrinterAddrButton Method

Executes the communication Read Printer Address button action. Illustrates use of BSReadPrinterAddress (see page 19) from the Bodystat API

C++

```
afx_msg void OnClickedComReadPrinterAddrButton();
```

C#

```
public afx_msg void OnClickedComReadPrinterAddrButton();
```

Visual Basic

```
Public Function OnClickedComReadPrinterAddrButton() As afx_msg void
```

Java

```
public afx_msg void OnClickedComReadPrinterAddrButton();
```

MATLAB

```
OnClickedComReadPrinterAddrButton
```

Body Source

```
void CBodystatMFCDlg::OnClickedComReadPrinterAddrButton()
{
 UpdateData(TRUE); // Grab the com port value from the edit box of the form

 CWaitCursor wc;  // Lengthy operation, show the user we are doing something

 // Open a com port to the device
 BSError bsErr = BSOpenComport(m_strComPort.GetBuffer(10), 10);
 if(bsErr == Bodystat::NoError)
 {
  // Connect to the device
  bsErr = BSConnect();
  if(bsErr == Bodystat::NoError)
  {
   CString strPrinterAddr;

   // Read the address of the printer the device is paired to for direct printing
   bsErr = BSReadPrinterAddress(strPrinterAddr.GetBuffer(16), 16);
   strPrinterAddr.ReleaseBuffer();
   if(bsErr == Bodystat::NoError)
   {
    CString strMsg;
    strMsg.Format(_T("Queried device successfully\nPrinter Address: %s"), strPrinterAddr);
    AfxMessageBox(strMsg, MB_ICONINFORMATION);
```

```
      }
      else
       AfxMessageBox(_T("Error querying printer address from device.\nNot all models support
  direct printing."));
    }
    else
     AfxMessageBox(_T("Error connecting to device."));

    BSCloseComport();
   }
   else
    AfxMessageBox(_T("Error opening com port."));
  }
```

4.1.1.1.3.14 CBodystatMFCDlg::OnClickedComReadProtocolInfoButton Method

Executes the communication ReadProtocolInfo button action. Illustrates use of BSReadProtocolInfo (see page 19) from the Bodystat API

C++

```
afx_msg void OnClickedComReadProtocolInfoButton();
```

C#

```
public afx_msg void OnClickedComReadProtocolInfoButton();
```

Visual Basic

```
Public Function OnClickedComReadProtocolInfoButton() As afx_msg void
```

Java

```
public afx_msg void OnClickedComReadProtocolInfoButton();
```

MATLAB

```
OnClickedComReadProtocolInfoButton
```

Body Source

```
void CBodystatMFCDlg::OnClickedComReadProtocolInfoButton()
{
 UpdateData(TRUE); // Grab the com port value from the edit box of the form

 CWaitCursor wc;  // Lengthy operation, show the user we are doing something

 // Open a com port to the device
 BSError bsErr = BSOpenComport(m_strComPort.GetBuffer(10), 10);
 if(bsErr == Bodystat::NoError)
 {
  // Connect to the device
  bsErr = BSConnect();
  if(bsErr == Bodystat::NoError)
  {
   BYTE byProtocolVersion;
   BYTE byDataVersion;
   BYTE byAuxInfo;

   // Read Protocol Information from the device
   bsErr = BSReadProtocolInfo(&byProtocolVersion, &byDataVersion, &byAuxInfo);
   if(bsErr == Bodystat::NoError)
   {
    CString strMsg;
    strMsg.Format(_T("Queried device successfully\nProtocol Version: %d\nData Version:
%d\nAux Info: %d"), byProtocolVersion, byDataVersion, byAuxInfo);
    AfxMessageBox(strMsg, MB_ICONINFORMATION);
   }
   else
    AfxMessageBox(_T("Error querying protocol information from device."));
  }
  else
   AfxMessageBox(_T("Error connecting to device."));

  BSCloseComport();
 }
 else
  AfxMessageBox(_T("Error opening com port."));
}
```

4.1.1.1.3.15 CBodystatMFCDlg::OnClickedComReadSerialNumberButton Method

Executes the communication Read Serial Number button action. Illustrates use of BSReadSerialNumber (see page 20) from the Bodystat API

C++

```
afx_msg void OnClickedComReadSerialNumberButton();
```

C#

```
public afx_msg void OnClickedComReadSerialNumberButton();
```

Visual Basic

```
Public Function OnClickedComReadSerialNumberButton() As afx_msg void
```

Java

```
public afx_msg void OnClickedComReadSerialNumberButton();
```

MATLAB

```
OnClickedComReadSerialNumberButton
```

Body Source

```
void CBodystatMFCDlg::OnClickedComReadSerialNumberButton()
{
 UpdateData(TRUE); // Grab the com port value from the edit box of the form

 CWaitCursor wc;  // Lengthy operation, show the user we are doing something

 // Open a com port to the device
 BSError bsErr = BSOpenComport(m_strComPort.GetBuffer(10), 10);
 if(bsErr == Bodystat::NoError)
 {
  // Connect to the device
  bsErr = BSConnect();
  if(bsErr == Bodystat::NoError)
  {
   ULONG ulSerialNumber;

   // Read the unique serial number from the device
   bsErr = BSReadSerialNumber(&ulSerialNumber);
   if(bsErr == Bodystat::NoError)
   {
    CString strMsg;
    strMsg.Format(_T("Queried device successfully\nSerial Number: %d"), ulSerialNumber);
    AfxMessageBox(strMsg, MB_ICONINFORMATION);
   }
   else
    AfxMessageBox(_T("Error querying serial number from device."));
  }
  else
   AfxMessageBox(_T("Error connecting to device."));

  BSCloseComport();
 }
 else
  AfxMessageBox(_T("Error opening com port."));
}
```

4.1.1.1.3.16 CBodystatMFCDlg::OnClickedComReadStoredTestDataButton Method

Executes the communication Read Stored Test Data button action. Illustrates use of BSReadStoredTestData (see page 21) from the Bodystat API to download data

C++

```
afx_msg void OnClickedComReadStoredTestDataButton();
```

C#

```
public afx_msg void OnClickedComReadStoredTestDataButton();
```

Visual Basic

```
Public Function OnClickedComReadStoredTestDataButton() As afx_msg void
```

Java

```
public afx_msg void OnClickedComReadStoredTestDataButton();
```

MATLAB

```
OnClickedComReadStoredTestDataButton
```

Body Source

```
void CBodystatMFCDlg::OnClickedComReadStoredTestDataButton()
{
 UpdateData(TRUE); // Grab the com port value from the edit box of the form

 CWaitCursor wc;  // Lengthy operation, show the user we are doing something

 // Open a com port to the device
 BSError bsErr = BSOpenComport(m_strComPort.GetBuffer(10), 10);
 if(bsErr == Bodystat::NoError)
 {
  // Connect to the device
  bsErr = BSConnect();
  if(bsErr == Bodystat::NoError)
  {
   // Set the array size in the raw data strucutre (SDK expects a fixed value here
currently)
   m_bsRawData.iRecordArraySize = Bodystat::BS_RAWDATA_ARRAYSIZE;

   // Download test measurement data from the device
   bsErr = BSReadStoredTestData(&m_bsRawData);
   if(bsErr == Bodystat::NoError)
   {
    CString strMsg;
    strMsg.Format(_T("Download complete\nRetrieved %d records\nFirst Test Number: %d"),
m_bsRawData.iTotalNumRecs, m_bsRawData.ulFirstTestNum);

    // Parse the data and place it in the measurement list control
    PopulateMeasurementList(&m_bsRawData);

    AfxMessageBox(strMsg, MB_ICONINFORMATION);
   }
   else
    AfxMessageBox(_T("Error downloading stored test data from the device."));
  }
  else
   AfxMessageBox(_T("Error connecting to device."));

  BSCloseComport();
 }
 else
  AfxMessageBox(_T("Error opening com port."));
}
```

4.1.1.1.3.17 CBodystatMFCDlg::OnClickedComReadTestMeasurementButton Method

Executes the communication Read Test Measurement button action. Performs an immediate test measurement on the device using the BSReadTestBodystat (see page 21) function from the Bodystat API

C++

```
afx_msg void OnClickedComReadTestMeasurementButton();
```

C#

```
public afx_msg void OnClickedComReadTestMeasurementButton();
```

Visual Basic

```
Public Function OnClickedComReadTestMeasurementButton() As afx_msg void
```

Java

```
public afx_msg void OnClickedComReadTestMeasurementButton();
```

MATLAB

```
OnClickedComReadTestMeasurementButton
```

Body Source

```
void CBodystatMFCDlg::OnClickedComReadTestMeasurementButton()
{
 UpdateData(TRUE); // Grab the com port value from the edit box of the form

 CWaitCursor wc;  // Lengthy operation, show the user we are doing something

 // Open a com port to the device
 BSError bsErr = BSOpenComport(m_strComPort.GetBuffer(10), 10);
 if(bsErr == Bodystat::NoError)
 {
  // Connect to the device
  bsErr = BSConnect();
  if(bsErr == Bodystat::NoError)
  {
   UINT iZ5, iZ50, iZ100, iZ200, iR50;
   float fX50, fPA50;

   // Perform the test measurement on the device and get the raw electrical bio-impedance
results
   bsErr = BSReadTestBodystat(&iZ5, &iZ50, &iZ100, &iZ200, &iR50, &fX50, &fPA50);
   if(bsErr == Bodystat::NoError)
   {
    CString strMsg;
    strMsg.Format(_T("Measurement complete\nImpedance 5kHz: %d\nImpedance 50kHz:
%d\nImpedance 100kHz: %d\nImpedance 200kHz: %d\nResistance 50kHz: %d\nReactance 50kHz:
%.1f\nPhase Angle 50kHz: %.1f"),
        iZ5, iZ50, iZ100, iZ200, iR50, fX50, fPA50);
    AfxMessageBox(strMsg, MB_ICONINFORMATION);
   }
   else
    AfxMessageBox(_T("Error performing test measurement on the device."));
  }
  else
   AfxMessageBox(_T("Error connecting to device."));

  BSCloseComport();
 }
 else
  AfxMessageBox(_T("Error opening com port."));
}
```

4.1.1.1.3.18 CBodystatMFCDlg::OnClickedComWriteCurrentTimeButton Method

Executes the communication Write Current Time button action. Illustrates use of BSWriteCurrentTime (see page 24) from the Bodystat API

C++

```
afx_msg void OnClickedComWriteCurrentTimeButton();
```

C#

```
public afx_msg void OnClickedComWriteCurrentTimeButton();
```

Visual Basic

```
Public Function OnClickedComWriteCurrentTimeButton() As afx_msg void
```

Java

```
public afx_msg void OnClickedComWriteCurrentTimeButton();
```

MATLAB

```
OnClickedComWriteCurrentTimeButton
```

Body Source

```
void CBodystatMFCDlg::OnClickedComWriteCurrentTimeButton()
{
 UpdateData(TRUE); // Grab the com port value from the edit box of the form

 CWaitCursor wc;  // Lengthy operation, show the user we are doing something

 // Open a com port to the device
 BSError bsErr = BSOpenComport(m_strComPort.GetBuffer(10), 10);
 if(bsErr == Bodystat::NoError)
 {
  // Connect to the device
  bsErr = BSConnect();
  if(bsErr == Bodystat::NoError)
  {
   // Write the current local PC time to the internal clock of the device
   bsErr = BSWriteCurrentTime();
   if(bsErr == Bodystat::NoError)
   {
    AfxMessageBox(_T("Device updated successfully\nCurrent date/time updated."),
MB_ICONINFORMATION);
   }
   else
    AfxMessageBox(_T("Error writing current date/time to device.\nNot all models have an
inbuilt real time clock."));
  }
  else
   AfxMessageBox(_T("Error connecting to device."));

  BSCloseComport();
 }
 else
  AfxMessageBox(_T("Error opening com port."));
}
```

4.1.1.1.3.19 CBodystatMFCDlg::OnClickedResultsButton Method

Executes the Results button action. Displays the results dialog with the selected test measurement or a sample result if none is selected.

C++

```
afx_msg void OnClickedResultsButton();
```

C#

```
public afx_msg void OnClickedResultsButton();
```

Visual Basic

```
Public Function OnClickedResultsButton() As afx_msg void
```

Java

```
public afx_msg void OnClickedResultsButton();
```

MATLAB

```
OnClickedResultsButton
```

Body Source

```
void CBodystatMFCDlg::OnClickedResultsButton()
{
 ShowResultsDialog();   // Display the selected result
}
```

4.1.1.1.3.20 CBodystatMFCDlg::OnDblclkTestMeasurementList Method

Executes when an item in the measurement list is double clicked.Displays the results dialog with the selected test measurement (or a sample result if none is selected.)

C++

```
afx_msg void OnDblclkTestMeasurementList(NMHDR * pNMHDR, LRESULT * pResult);
```

C#

```
public afx_msg void OnDblclkTestMeasurementList(ref NMHDR pNMHDR, ref LRESULT pResult);
```

Visual Basic

```
Public Function OnDblclkTestMeasurementList(ByRef pNMHDR As NMHDR, ByRef pResult As
LRESULT) As afx_msg void
```

Java

```
public afx_msg void OnDblclkTestMeasurementList(NMHDR pNMHDR, LRESULT pResult);
```

MATLAB

```
OnDblclkTestMeasurementList
```

Body Source

```
void CBodystatMFCDlg::OnDblclkTestMeasurementList(NMHDR *pNMHDR, LRESULT *pResult)
{
 ShowResultsDialog();   // Display the selected result
 *pResult = 0;
}
```

4.1.1.1.3.21 CBodystatMFCDlg::OnInitDialog Method

C++

```
virtual BOOL OnInitDialog();
```

C#

```
protected BOOL OnInitDialog();
```

Visual Basic

```
Protected Function OnInitDialog() As BOOL
```

Java

```
protected BOOL OnInitDialog();
```

MATLAB

```
OnInitDialog
```

Description

Generated message map functions

CBodystatMFCDlg (see page 156) message handlers

Body Source

```
BOOL CBodystatMFCDlg::OnInitDialog()
{
 CDialogEx::OnInitDialog();

 // Add "About..." menu item to system menu.

 // IDM_ABOUTBOX must be in the system command range.
 ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
 ASSERT(IDM_ABOUTBOX < 0xF000);

 CMenu* pSysMenu = GetSystemMenu(FALSE);
 if (pSysMenu != NULL)
 {
  BOOL bNameValid;
```

```
    CString strMenuTitle;

    // Add About menu
    bNameValid = strMenuTitle.LoadString(IDS_ABOUTBOX);
    ASSERT(bNameValid);
    if (!strMenuTitle.IsEmpty())
    {
     pSysMenu->AppendMenu(MF_SEPARATOR);
     pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strMenuTitle);   // Add about menu item to
  the main file menu
    }
   }

   // Set the icon for this dialog.  The framework does this automatically
   //  when the application's main window is not a dialog
   SetIcon(m_hIcon, TRUE);   // Set big icon
   SetIcon(m_hIcon, FALSE);  // Set small icon

   // Add Vista Shield icon to some buttons which may require elevated user privileges
   Button_SetElevationRequiredState(m_btnAutoSetup, TRUE);
   Button_SetElevationRequiredState(m_btnAuthenticate, TRUE);
   Button_SetElevationRequiredState(m_btnUnauthenticate, TRUE);

   // Add columns to the test measurement list ready for download of data
   int j=0;
   m_TestMeasurementListCtrl.InsertColumn(j++, _T("Test No"), 0, 50);
   m_TestMeasurementListCtrl.InsertColumn(j++, _T("Test Date/Time"), 0, 130);
   m_TestMeasurementListCtrl.InsertColumn(j++, _T("Model"), 0, 140);
   m_TestMeasurementListCtrl.InsertColumn(j++, _T("Serial No"), 0, 75);
   m_TestMeasurementListCtrl.InsertColumn(j++, _T("Gender"), 0, 75);
   m_TestMeasurementListCtrl.InsertColumn(j++, _T("Age"), 0, 75);
   m_TestMeasurementListCtrl.InsertColumn(j++, _T("Height"), 0, 75);
   m_TestMeasurementListCtrl.InsertColumn(j++, _T("Weight"), 0, 75);
   m_TestMeasurementListCtrl.InsertColumn(j++, _T("Activity"), 0, 75);
   m_TestMeasurementListCtrl.InsertColumn(j++, _T("Waist"), 0, 75);
   m_TestMeasurementListCtrl.InsertColumn(j++, _T("Hip"), 0, 75);
   m_TestMeasurementListCtrl.InsertColumn(j++, _T("Imp 5 kHz"), 0, 75);
   m_TestMeasurementListCtrl.InsertColumn(j++, _T("Imp 50 kHz"), 0, 75);
   m_TestMeasurementListCtrl.InsertColumn(j++, _T("Imp 100 kHz"), 0, 75);
   m_TestMeasurementListCtrl.InsertColumn(j++, _T("Imp 200 kHz"), 0, 75);
   m_TestMeasurementListCtrl.InsertColumn(j++, _T("Res 50 kHz"), 0, 75);
   m_TestMeasurementListCtrl.InsertColumn(j++, _T("Reac 50 kHz"), 0, 75);
   m_TestMeasurementListCtrl.InsertColumn(j++, _T("Phase 50 kHz"), 0, 80);
   m_TestMeasurementListCtrl.SetExtendedStyle( m_TestMeasurementListCtrl.GetExtendedStyle() |
  LVS_EX_FULLROWSELECT );

   return TRUE;  // return TRUE  unless you set the focus to a control
  }
```

4.1.1.1.3.22 CBodystatMFCDlg::OnPaint Method

C++

```
afx_msg void OnPaint();
```

C#

```
protected afx_msg void OnPaint();
```

Visual Basic

```
Protected Function OnPaint() As afx_msg void
```

Java

```
protected afx_msg void OnPaint();
```

MATLAB

```
OnPaint
```

Description

If you add a minimize button to your dialog, you will need the code below to draw the icon. For MFC applications using the document/view model, this is automatically done for you by the framework.

Body Source

```
void CBodystatMFCDlg::OnPaint()
{
 if (IsIconic())
 {
  CPaintDC dc(this); // device context for painting

  SendMessage(WM_ICONERASEBKGND, reinterpret_cast<WPARAM>(dc.GetSafeHdc()), 0);

  // Center icon in client rectangle
  int cxIcon = GetSystemMetrics(SM_CXICON);
  int cyIcon = GetSystemMetrics(SM_CYICON);
  CRect rect;
  GetClientRect(&rect);
  int x = (rect.Width() - cxIcon + 1) / 2;
  int y = (rect.Height() - cyIcon + 1) / 2;

  // Draw the icon
  dc.DrawIcon(x, y, m_hIcon);
 }
 else
 {
  CDialogEx::OnPaint();
 }
}
```

4.1.1.1.3.23 CBodystatMFCDlg::OnQueryDragIcon Method

C++

```
afx_msg HCURSOR OnQueryDragIcon();
```

C#

```
protected afx_msg HCURSOR OnQueryDragIcon();
```

Visual Basic

```
Protected Function OnQueryDragIcon() As afx_msg HCURSOR
```

Java

```
protected afx_msg HCURSOR OnQueryDragIcon();
```

MATLAB

```
OnQueryDragIcon
```

Description

The system calls this function to obtain the cursor to display while the user drags the minimized window.

Body Source

```
HCURSOR CBodystatMFCDlg::OnQueryDragIcon()
{
 return static_cast<HCURSOR>(m_hIcon);
}
```

4.1.1.1.3.24 CBodystatMFCDlg::OnSysCommand Method

C++

```
afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
```

C#

```
protected afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
```

Visual Basic

```
Protected Function OnSysCommand(nID As UINT, lParam As LPARAM) As afx_msg void
```

Java

```
protected afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
```

MATLAB

```
OnSysCommand
```

Description

This is OnSysCommand, a member of class CBodystatMFCDlg.

Body Source

```
void CBodystatMFCDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
 if ((nID & 0xFFF0) == IDM_ABOUTBOX)
 {
  CAboutDlg dlgAbout;
  dlgAbout.DoModal();
 }
 else
 {
  CDialogEx::OnSysCommand(nID, lParam);
 }
}
```

4.1.1.1.3.25 CBodystatMFCDlg::PopulateMeasurementList Method

Populate the measurement list control with test measurement data from the API. Illustrates iteration through a BSRawData (see page 40) structure, working with data in a BSMeasurement (see page 36) structure and helper functions from the Bodystat API like BSGetDeviceModelName (see page 14)

C++

```
void PopulateMeasurementList(BSRawData * pRawData);
```

C#

```
public PopulateMeasurementList(ref BSRawData pRawData);
```

Visual Basic

```
Public Sub PopulateMeasurementList(ByRef pRawData As BSRawData)
```

Java

```
public PopulateMeasurementList(BSRawData pRawData);
```

MATLAB

```
PopulateMeasurementList
```

Parameters

Parameters	Description
BSRawData * pRawData	[in] RawData to be parsed.

Body Source

```
void CBodystatMFCDlg::PopulateMeasurementList(BSRawData *pRawData)
{
 // Clear any previous items in the list
 m_TestMeasurementListCtrl.DeleteAllItems();

 // Check we have some raw data to work with
 if(pRawData)
 {
  CString strItem;
  int nItem;

  // Process each record in the structure
  for(int i=0; i<pRawData->iTotalNumRecs; i++)
  {
   int j=1; // Column number

   // Test no
   strItem.Format(_T("%d"), i + pRawData->ulFirstTestNum);
   nItem = m_TestMeasurementListCtrl.InsertItem(i, strItem);
```

```
    m_TestMeasurementListCtrl.SetItemData(nItem, (DWORD_PTR)&(pRawData->record[i]));

    // Test date/time
    COleDateTime dtTestDate(pRawData->record[i].tTestDate);
    m_TestMeasurementListCtrl.SetItemText(nItem, j++, dtTestDate.Format());

    // Device model
    BSGetDeviceModelName(pRawData->record[i].iDeviceModel, strItem.GetBuffer(32), 32);
    strItem.ReleaseBuffer();
    m_TestMeasurementListCtrl.SetItemText(nItem, j++, strItem);

    // Device serial number
    strItem.Format(_T("%d"), pRawData->record[i].ulDeviceSerialNumber);
    m_TestMeasurementListCtrl.SetItemText(nItem, j++, strItem);

    // Gender
    m_TestMeasurementListCtrl.SetItemText(nItem, j++, pRawData->record[i].iGender ==
  Bodystat::BSMale ? _T("Male") : _T("Female"));

    // Age
    strItem.Format(_T("%d"), pRawData->record[i].iAge);
    m_TestMeasurementListCtrl.SetItemText(nItem, j++, strItem);

    // Height
    strItem.Format(_T("%d cm"), pRawData->record[i].iHeight);
    m_TestMeasurementListCtrl.SetItemText(nItem, j++, strItem);

    // Weight
    strItem.Format(_T("%.1f kg"), pRawData->record[i].fWeight);
    m_TestMeasurementListCtrl.SetItemText(nItem, j++, strItem);

    // Activity
    switch(pRawData->record[i].iActivity)
    {
     case 1:  strItem = _T("Very Low");  break;
     case 2:  strItem = _T("Low/Medium");  break;
     case 3:  strItem = _T("Medium");   break;
     case 4:  strItem = _T("Medium/High"); break;
     case 5:  strItem = _T("Very High");  break;
     default: strItem = _T("Unknown");  break;
    }
    m_TestMeasurementListCtrl.SetItemText(nItem, j++, strItem);

    // Waist
    strItem.Format(_T("%d cm"), pRawData->record[i].iWaist);
    m_TestMeasurementListCtrl.SetItemText(nItem, j++, strItem);

    // Hip
    strItem.Format(_T("%d cm"), pRawData->record[i].iHip);
    m_TestMeasurementListCtrl.SetItemText(nItem, j++, strItem);

    // Z5 (Impedance at 5 kHz)
    strItem.Format(_T("%d ohm"), pRawData->record[i].iZ_5kHz);
    m_TestMeasurementListCtrl.SetItemText(nItem, j++, strItem);

    // Z50 (Impedance at 50 kHz)
    strItem.Format(_T("%d ohm"), pRawData->record[i].iZ_50kHz);
    m_TestMeasurementListCtrl.SetItemText(nItem, j++, strItem);

    // Z100 (Impedance at 100 kHz)
    strItem.Format(_T("%d ohm"), pRawData->record[i].iZ_100kHz);
    m_TestMeasurementListCtrl.SetItemText(nItem, j++, strItem);

    // Z200 (Impedance at 200 kHz)
    strItem.Format(_T("%d ohm"), pRawData->record[i].iZ_200kHz);
    m_TestMeasurementListCtrl.SetItemText(nItem, j++, strItem);

    // R50 (Resistance at 50 kHz)
    strItem.Format(_T("%d ohm"), pRawData->record[i].iR_50kHz);
    m_TestMeasurementListCtrl.SetItemText(nItem, j++, strItem);

    // X50 (Reactance at 50 kHz)
```

```
     strItem.Format(_T("%.1f ohm"), pRawData->record[i].fX_50kHz);
     m_TestMeasurementListCtrl.SetItemText(nItem, j++, strItem);

     // PA50 (Phase Angle at 50 kHz)
     strItem.Format(_T("%.1f deg"), pRawData->record[i].fPA_50kHz);
     m_TestMeasurementListCtrl.SetItemText(nItem, j++, strItem);

    }
   }
  }
```

4.1.1.1.3.26 CBodystatMFCDlg::ShowResultsDialog Method

Displays the results dialog with the selected test measurement (or a sample result if none is selected.)

C++

```
void ShowResultsDialog();
```

C#

```
public ShowResultsDialog();
```

Visual Basic

```
Public Sub ShowResultsDialog()
```

Java

```
public ShowResultsDialog();
```

MATLAB

```
ShowResultsDialog
```

Body Source

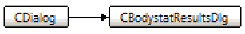
```
void CBodystatMFCDlg::ShowResultsDialog()
{
 CBodystatResultsDlg resultsDlg;  // Initialise the results dialog

 // Populate the results dialog with the details of the selected measurement
 // If no item is selected, we simply show the results dialog with its default values
 int nItem = m_TestMeasurementListCtrl.GetNextItem( -1, LVNI_SELECTED );
 if(nItem >= 0)
 {
  BSMeasurement *pBSM = (BSMeasurement *)m_TestMeasurementListCtrl.GetItemData(nItem);
  if(pBSM)
  {
   resultsDlg.m_iModel = pBSM->iDeviceModel;
   resultsDlg.m_dtTestDate = pBSM->tTestDate;
   resultsDlg.m_iGender = pBSM->iGender == BSMale ? 0 : 1;
   resultsDlg.m_iAge = pBSM->iAge;
   resultsDlg.m_iHeight = pBSM->iHeight;
   resultsDlg.m_fWeight  = pBSM->fWeight;
   resultsDlg.m_iActivityLevel = pBSM->iActivity;
   resultsDlg.m_iWaist = pBSM->iWaist;
   resultsDlg.m_iHip = pBSM->iHip;

   resultsDlg.m_bModeBodyComposition = TRUE;
   resultsDlg.m_bModeHydration = TRUE;

   resultsDlg.m_iImpedance5K = pBSM->iZ_5kHz;
   resultsDlg.m_iImpedance50K = pBSM->iZ_50kHz;
   resultsDlg.m_iImpedance100K = pBSM->iZ_100kHz;
   resultsDlg.m_iImpedance200K = pBSM->iZ_200kHz;
   resultsDlg.m_iResistance50K = pBSM->iR_50kHz;
   resultsDlg.m_fReactance50K = pBSM->fX_50kHz;
   resultsDlg.m_fPhaseAngle50K = pBSM->fPA_50kHz;
  }
 }
 else
  AfxMessageBox(_T("No measurement selected, an example measurement will be shown
instead."), MB_ICONINFORMATION);
```

```
  // Show the results dialog
  resultsDlg.DoModal();
}
```

4.1.1.2 CBodystatResultsDlg Class

CBodystatResultsDlg. Dialog for viewing the Bodystat test measurement results and normals.

Class Hierarchy



C++

```
class CBodystatResultsDlg : public CDialog;
```

C#

```
public class CBodystatResultsDlg : CDialog;
```

Visual Basic

```
Public Class CBodystatResultsDlg
Inherits CDialog
```

Java

```
public class CBodystatResultsDlg CDialog;
```

MATLAB

```
CBodystatResultsDlg
```

File

BodystatResultsDlg.h

Methods

	Name	Description
◆ ▽	~CBodystatResultsDlg (↗ see page 186)	This is ~CBodystatResultsDlg, a member of class CBodystatResultsDlg.
◆	CBodystatResultsDlg (↗ see page 186)	CBodystatResultsDlg Constructor.

CBodystatResultsDlg Data Members

	Name	Description
◆	m_bModeBodyComposition (↗ see page 187)	This is m_bModeBodyComposition, a member of class CBodystatResultsDlg.
◆	m_bModeHydration (↗ see page 188)	This is m_bModeHydration, a member of class CBodystatResultsDlg.
◆	m_dtTestDate (↗ see page 188)	This is m_dtTestDate, a member of class CBodystatResultsDlg.
◆	m_fPhaseAngle50K (↗ see page 188)	This is m_fPhaseAngle50K, a member of class CBodystatResultsDlg.
◆	m_fReactance50K (↗ see page 189)	This is m_fReactance50K, a member of class CBodystatResultsDlg.
◆	m_fWeight (↗ see page 189)	This is m_fWeight, a member of class CBodystatResultsDlg.
◆	m_iActivityLevel (↗ see page 189)	This is m_iActivityLevel, a member of class CBodystatResultsDlg.
◆	m_iAge (↗ see page 190)	This is m_iAge, a member of class CBodystatResultsDlg.
◆	m_iGender (↗ see page 190)	This is m_iGender, a member of class CBodystatResultsDlg.
◆	m_iHeight (↗ see page 190)	This is m_iHeight, a member of class CBodystatResultsDlg.
◆	m_iHip (↗ see page 191)	This is m_iHip, a member of class CBodystatResultsDlg.
◆	m_iImpedance100K (↗ see page 191)	This is m_iImpedance100K, a member of class CBodystatResultsDlg.

	m_iImpedance200K (see page 191)	This is m_iImpedance200K, a member of class CBodystatResultsDlg.
	m_iImpedance50K (see page 192)	This is m_iImpedance50K, a member of class CBodystatResultsDlg.
	m_iImpedance5K (see page 192)	This is m_iImpedance5K, a member of class CBodystatResultsDlg.
	m_iModel (see page 192)	This is m_iModel, a member of class CBodystatResultsDlg.
	m_iResistance50K (see page 193)	This is m_iResistance50K, a member of class CBodystatResultsDlg.
	m_iWaist (see page 193)	This is m_iWaist, a member of class CBodystatResultsDlg.
	m_ModelComboCtrl (see page 193)	This is m_ModelComboCtrl, a member of class CBodystatResultsDlg.
	m_ResultsListCtrl (see page 194)	This is m_ResultsListCtrl, a member of class CBodystatResultsDlg.

CBodystatResultsDlg Methods

	Name	Description
	CopyToClipboard (see page 194)	Copies measurement parameters, results and normals to the clipboard. Useful if you need to make note of a particular test set for diagnostics.
	DoDataExchange (see page 196)	DDX/DDV support Bind the form controls to member variables and set basic input range limits
	GetSystemErrorMessage (see page 197)	Gets a system error message.
	OnClickedCalculateResultsButton (see page 198)	Executes the Calculate Results button action. Uses the Bodystat API to calculate the results and normals for this test measurement.
	OnClickedCopyToClipboardButton (see page 200)	Executes the Copy To Clipboard button action.
	OnInitDialog (see page 200)	CBodystatResultsDlg message handlers
	PopulateListCtrl (see page 201)	Populate the list control with results and normals.

4.1.1.2.1 CBodystatResultsDlg::~CBodystatResultsDlg Destructor

C++

```
virtual ~CBodystatResultsDlg();
```

C#

```
public ~CBodystatResultsDlg();
```

Visual Basic

```
Public Sub Finalize()
```

Java

```
public finalize();
```

MATLAB

```
~CBodystatResultsDlg
```

Description

This is ~CBodystatResultsDlg, a member of class CBodystatResultsDlg.

Body Source

```
CBodystatResultsDlg::~CBodystatResultsDlg()
{
}
```

4.1.1.2.2 CBodystatResultsDlg::CBodystatResultsDlg Constructor

CBodystatResultsDlg Constructor.

C++

```
CBodystatResultsDlg(CWnd* pParent = NULL);
```

C#

```
public CBodystatResultsDlg(ref CWnd pParent);
```

Visual Basic

```
Public Sub New(ByRef pParent As CWnd = NULL)
```

Java

```
public CBodystatResultsDlg(CWnd pParent);
```

MATLAB

```
CBodystatResultsDlg
```

Parameters

Parameters	Description
CWnd* pParent = NULL	=NULL

Description

standard constructor

Body Source

```
CBodystatResultsDlg::CBodystatResultsDlg(CWnd* pParent /*=NULL*/)
 : CDialog(CBodystatResultsDlg::IDD, pParent)
{
 // Setup the form with some sensible default values
 m_iModel = (int)BSQuadScan_BT;
 m_iGender = 0;
 m_iAge = 30;
 m_iHeight = 165;
 m_fWeight = 85.0f;
 m_iActivityLevel = 3;
 m_iWaist = 85;
 m_iHip = 90;

 m_bModeBodyComposition = TRUE;
 m_bModeHydration = TRUE;

 m_iImpedance5K = 500;
 m_iImpedance50K = 500;
 m_iImpedance100K = 500;
 m_iImpedance200K = 500;
 m_iResistance50K = 500;
 m_fReactance50K = 52.0;
 m_fPhaseAngle50K = (float)6.1;
}
```

4.1.1.2.3 CBodystatResultsDlg Data Members

4.1.1.2.3.1 CBodystatResultsDlg::m_bModeBodyComposition Data Member

C++

```
BOOL m_bModeBodyComposition;
```

C#

```
public BOOL m_bModeBodyComposition;
```

Visual Basic

```
Public m_bModeBodyComposition As BOOL
```

Java

```
public BOOL m_bModeBodyComposition;
```

MATLAB

```
m_bModeBodyComposition
```

Description

This is m_bModeBodyComposition, a member of class CBodystatResultsDlg.

4.1.1.2.3.2 CBodystatResultsDlg::m_bModeHydration Data Member

C++

```
BOOL m_bModeHydration;
```

C#

```
public BOOL m_bModeHydration;
```

Visual Basic

```
Public m_bModeHydration As BOOL
```

Java

```
public BOOL m_bModeHydration;
```

MATLAB

```
m_bModeHydration
```

Description

This is m_bModeHydration, a member of class CBodystatResultsDlg.

4.1.1.2.3.3 CBodystatResultsDlg::m_dtTestDate Data Member

C++

```
COleDateTime m_dtTestDate;
```

C#

```
public COleDateTime m_dtTestDate;
```

Visual Basic

```
Public m_dtTestDate As COleDateTime
```

Java

```
public COleDateTime m_dtTestDate;
```

MATLAB

```
m_dtTestDate
```

Description

This is m_dtTestDate, a member of class CBodystatResultsDlg.

4.1.1.2.3.4 CBodystatResultsDlg::m_fPhaseAngle50K Data Member

C++

```
float m_fPhaseAngle50K;
```

C#

```
public float m_fPhaseAngle50K;
```

Visual Basic

```
Public m_fPhaseAngle50K As float
```

Java

```
public float m_fPhaseAngle50K;
```

MATLAB

```
m_fPhaseAngle50K
```

Description

This is m_fPhaseAngle50K, a member of class CBodystatResultsDlg.

4.1.1.2.3.5 CBodystatResultsDlg::m_fReactance50K Data Member

C++

```
float m_fReactance50K;
```

C#

```
public float m_fReactance50K;
```

Visual Basic

```
Public m_fReactance50K As float
```

Java

```
public float m_fReactance50K;
```

MATLAB

```
m_fReactance50K
```

Description

This is m_fReactance50K, a member of class CBodystatResultsDlg.

4.1.1.2.3.6 CBodystatResultsDlg::m_fWeight Data Member

C++

```
float m_fWeight;
```

C#

```
public float m_fWeight;
```

Visual Basic

```
Public m_fWeight As float
```

Java

```
public float m_fWeight;
```

MATLAB

```
m_fWeight
```

Description

This is m_fWeight, a member of class CBodystatResultsDlg.

4.1.1.2.3.7 CBodystatResultsDlg::m_iActivityLevel Data Member

C++

```
int m_iActivityLevel;
```

C#

```
public int m_iActivityLevel;
```

Visual Basic

```
Public m_iActivityLevel As Integer
```

Java

```
public int m_iActivityLevel;
```

MATLAB

```
m_iActivityLevel
```

Description

This is m_iActivityLevel, a member of class CBodystatResultsDlg.

4.1.1.2.3.8 CBodystatResultsDlg::m_iAge Data Member

C++

```
int m_iAge;
```

C#

```
public int m_iAge;
```

Visual Basic

```
Public m_iAge As Integer
```

Java

```
public int m_iAge;
```

MATLAB

```
m_iAge
```

Description

This is m_iAge, a member of class CBodystatResultsDlg.

4.1.1.2.3.9 CBodystatResultsDlg::m_iGender Data Member

C++

```
int m_iGender;
```

C#

```
public int m_iGender;
```

Visual Basic

```
Public m_iGender As Integer
```

Java

```
public int m_iGender;
```

MATLAB

```
m_iGender
```

Description

This is m_iGender, a member of class CBodystatResultsDlg.

4.1.1.2.3.10 CBodystatResultsDlg::m_iHeight Data Member

C++

```
int m_iHeight;
```

C#

```
public int m_iHeight;
```

Visual Basic

```
Public m_iHeight As Integer
```

Java

```
public int m_iHeight;
```

MATLAB

```
m_iHeight
```

Description

This is m_iHeight, a member of class CBodystatResultsDlg.

4.1.1.2.3.11 CBodystatResultsDlg::m_iHip Data Member

C++

```
int m_iHip;
```

C#

```
public int m_iHip;
```

Visual Basic

```
Public m_iHip As Integer
```

Java

```
public int m_iHip;
```

MATLAB

```
m_iHip
```

Description

This is m_iHip, a member of class CBodystatResultsDlg.

4.1.1.2.3.12 CBodystatResultsDlg::m_iImpedance100K Data Member

C++

```
int m_iImpedance100K;
```

C#

```
public int m_iImpedance100K;
```

Visual Basic

```
Public m_iImpedance100K As Integer
```

Java

```
public int m_iImpedance100K;
```

MATLAB

```
m_iImpedance100K
```

Description

This is m_iImpedance100K, a member of class CBodystatResultsDlg.

4.1.1.2.3.13 CBodystatResultsDlg::m_iImpedance200K Data Member

C++

```
int m_iImpedance200K;
```

C#

```
public int m_iImpedance200K;
```

Visual Basic

```
Public m_iImpedance200K As Integer
```

Java

```
public int m_iImpedance200K;
```

MATLAB

```
m_iImpedance200K
```

Description

This is m_iImpedance200K, a member of class CBodystatResultsDlg.

4.1.1.2.3.14 CBodystatResultsDlg::m_iImpedance50K Data Member

C++

```
int m_iImpedance50K;
```

C#

```
public int m_iImpedance50K;
```

Visual Basic

```
Public m_iImpedance50K As Integer
```

Java

```
public int m_iImpedance50K;
```

MATLAB

```
m_iImpedance50K
```

Description

This is m_iImpedance50K, a member of class CBodystatResultsDlg.

4.1.1.2.3.15 CBodystatResultsDlg::m_iImpedance5K Data Member

C++

```
int m_iImpedance5K;
```

C#

```
public int m_iImpedance5K;
```

Visual Basic

```
Public m_iImpedance5K As Integer
```

Java

```
public int m_iImpedance5K;
```

MATLAB

```
m_iImpedance5K
```

Description

This is m_iImpedance5K, a member of class CBodystatResultsDlg.

4.1.1.2.3.16 CBodystatResultsDlg::m_iModel Data Member

C++

```
int m_iModel;
```

C#

```
public int m_iModel;
```

Visual Basic

```
Public m_iModel As Integer
```

Java

```
public int m_iModel;
```

MATLAB

```
m_iModel
```

Description

This is m_iModel, a member of class CBodystatResultsDlg.

4.1.1.2.3.17 CBodystatResultsDlg::m_iResistance50K Data Member

C++

```
int m_iResistance50K;
```

C#

```
public int m_iResistance50K;
```

Visual Basic

```
Public m_iResistance50K As Integer
```

Java

```
public int m_iResistance50K;
```

MATLAB

```
m_iResistance50K
```

Description

This is m_iResistance50K, a member of class CBodystatResultsDlg.

4.1.1.2.3.18 CBodystatResultsDlg::m_iWaist Data Member

C++

```
int m_iWaist;
```

C#

```
public int m_iWaist;
```

Visual Basic

```
Public m_iWaist As Integer
```

Java

```
public int m_iWaist;
```

MATLAB

```
m_iWaist
```

Description

This is m_iWaist, a member of class CBodystatResultsDlg.

4.1.1.2.3.19 CBodystatResultsDlg::m_ModelComboCtrl Data Member

C++

```
CComboBox m_ModelComboCtrl;
```

C#

```
protected CComboBox m_ModelComboCtrl;
```

Visual Basic

```
Protected m_ModelComboCtrl As CComboBox
```

Java

```
protected CComboBox m_ModelComboCtrl;
```

MATLAB

```
m_ModelComboCtrl
```

Description

This is m_ModelComboCtrl, a member of class CBodystatResultsDlg.

4.1.1.2.3.20 CBodystatResultsDlg::m_ResultsListCtrl Data Member

C++

```
CListCtrl m_ResultsListCtrl;
```

C#

```
protected CListCtrl m_ResultsListCtrl;
```

Visual Basic

```
Protected m_ResultsListCtrl As CListCtrl
```

Java

```
protected CListCtrl m_ResultsListCtrl;
```

MATLAB

```
m_ResultsListCtrl
```

Description

This is m_ResultsListCtrl, a member of class CBodystatResultsDlg.

4.1.1.2.4 CBodystatResultsDlg Methods

4.1.1.2.4.1 CBodystatResultsDlg::CopyToClipboard Method

Copies measurement parameters, results and normals to the clipboard. Useful if you need to make note of a particular test set for diagnostics.

C++

```
void CopyToClipboard();
```

C#

```
protected CopyToClipboard();
```

Visual Basic

```
Protected Sub CopyToClipboard()
```

Java

```
protected CopyToClipboard();
```

MATLAB

```
CopyToClipboard
```

Body Source

```
void CBodystatResultsDlg::CopyToClipboard()
{
 CString strErrorMsg, strBuffer, strTemp;

 // Copy measurement input parameters to the clipboard
 strBuffer += _T("Bodystat Data\r\n");
 strBuffer += _T("-------------\r\n");

 CString strModel;
 BSGetDeviceModelName((BSDeviceModel)m_iModel, strModel.GetBuffer(32), 32);
 strModel.ReleaseBuffer();
 strTemp.Format(_T("Model:\t%s\r\n"), strModel);      strBuffer += strTemp;
 strTemp.Format(_T("Test Date/Time:\t%s\r\n"), m_dtTestDate.m_status == COleDateTime::null
? m_dtTestDate.Format() : _T("Unknown"));  strBuffer += strTemp;
 strTemp.Format(_T("Gender:\t%s\r\n"), m_iGender == 0 ? _T("Male") : _T("Female"));
strBuffer += strTemp;
 strTemp.Format(_T("Age:\t%d\tyears\r\n"), m_iAge);     strBuffer += strTemp;
 strTemp.Format(_T("Height:\t%d\tcm\r\n"), m_iHeight);    strBuffer += strTemp;
 strTemp.Format(_T("Weight:\t%.1f\tkg\r\n"), m_fWeight);    strBuffer += strTemp;
 strTemp.Format(_T("Activity Level:\t%d\r\n"), m_iActivityLevel); strBuffer += strTemp;
 strTemp.Format(_T("Waist:\t%d\tcm\r\n"), m_iWaist);     strBuffer += strTemp;
 strTemp.Format(_T("Hip:\t%d\tcm\r\n"), m_iHip);      strBuffer += strTemp;

 strTemp.Format(_T("Body Composition Mode:\t%s\r\n"),m_bModeBodyComposition ? _T("Enabled")
: _T("Disabled")); strBuffer += strTemp;
 strTemp.Format(_T("Hydration Mode:\t%s\r\n"),m_bModeHydration ? _T("Enabled") :
_T("Disabled"));   strBuffer += strTemp;;

 strTemp.Format(_T("Impedance 5 kHz:\t%d\tohm\r\n"), m_iImpedance5K);   strBuffer +=
strTemp;
 strTemp.Format(_T("Impedance 50 kHz:\t%d\tohm\r\n"), m_iImpedance50K);  strBuffer +=
strTemp;
 strTemp.Format(_T("Impedance 100 kHz:\t%d\tohm\r\n"), m_iImpedance100K);  strBuffer +=
strTemp;
 strTemp.Format(_T("Impedance 200 kHz:\t%d\tohm\r\n"), m_iImpedance200K);  strBuffer +=
strTemp;
 strTemp.Format(_T("Resistance 50 kHz:\t%d\tohm\r\n"), m_iResistance50K);  strBuffer +=
strTemp;
 strTemp.Format(_T("Reactance 50 kHz:\t%.1f\tohm\r\n"), m_fReactance50K);  strBuffer +=
strTemp;
 strTemp.Format(_T("Phase Angle 50 kHz:\t%.1f\t°\r\n"), m_fPhaseAngle50K); strBuffer +=
strTemp;

 strBuffer += _T("\r\n");

 // Copy the results and normals to the clipboard from the results list control
 int iRowCount = m_ResultsListCtrl.GetItemCount();
 int  iColCount = 0;
 CHeaderCtrl* pHeaderCtrl = m_ResultsListCtrl.GetHeaderCtrl();
 if (pHeaderCtrl != NULL)
 {
  HDITEM hdi;
  const int HDR_TEXT_SIZE = 256;
  TCHAR  lpBuffer[HDR_TEXT_SIZE];
  hdi.mask = HDI_TEXT;
  hdi.pszText = lpBuffer;
  hdi.cchTextMax = HDR_TEXT_SIZE;

  // Copy column titles
  for(int iDashed=0; iDashed < 2; iDashed++)         // Iterate through column titles
twice, once for the title, once to underline it with dashes
  {
   iColCount = pHeaderCtrl->GetItemCount();
   for(int i=0; i < iColCount; i++)
   {
    if(pHeaderCtrl->GetItem(i, &hdi))
    {
     if(hdi.mask & HDI_TEXT && hdi.pszText != LPSTR_TEXTCALLBACK)
     {
      strTemp = CString(lpBuffer);
      if(iDashed == 0)
       strBuffer += strTemp;         // Use the title
      else
       strBuffer += CString(_T('-'), strTemp.GetLength());  // Underline the title with
dashes
     }
    }
    strBuffer += '\t';
   }
   strBuffer += "\r\n";
  }
 }
 // Build clipboard text from the list control contents (results and normals)
 for (int i=0; i < iRowCount; i++)
 {
  for (int j=0; j < iColCount; j++)
   strBuffer += m_ResultsListCtrl.GetItemText(i, j) + '\t';

  strBuffer += "\r\n";
 }

 // Prepare the clipboard
 if (!OpenClipboard())
 {
  AfxMessageBox(_T("The clipboard is temporarily unavailable"));
  return;
 }
 if (!EmptyClipboard())
 {
  CloseClipboard();
  AfxMessageBox(_T("The clipboard cannot be emptied"));
  return;
 }

 // Allocate clipboard buffer
 HGLOBAL hGlobal;
 hGlobal = GlobalAlloc(GHND|GMEM_SHARE,  (strBuffer.GetLength()+1)* sizeof (TCHAR));
 if (!hGlobal)
 {
  CloseClipboard();
  AfxMessageBox(CString("Error allocating memory for clipboard data: ") +
GetSystemErrorMessage(GetLastError()));
  return;
 }

 // Prevent buffer from moving around and copy the text to it
 TCHAR * szClipboard = (TCHAR *)GlobalLock(hGlobal);
 if(szClipboard)
 {
  StringCchCopy(szClipboard,  (strBuffer.GetLength()+1)* sizeof (TCHAR), strBuffer);
  GlobalUnlock(hGlobal);
 }

 // Determine clipboard format (Unicode if available)
#ifdef UNICODE
 UINT uFmt = CF_UNICODETEXT;
#else
 UINT uFmt = CF_TEXT;
#endif

 // Assign clipboard buffer to the clipboard
 if (!SetClipboardData(uFmt, hGlobal))
 {
  AfxMessageBox(CString("Error copying to the clipboard: ") +
GetSystemErrorMessage(GetLastError()));
 }
 CloseClipboard();
}
```

4.1.1.2.4.2 CBodystatResultsDlg::DoDataExchange Method

C++

```
virtual void DoDataExchange(CDataExchange* pDX);
```

C#

```
protected DoDataExchange(ref CDataExchange pDX);
```

Visual Basic

```
Protected Sub DoDataExchange(ByRef pDX As CDataExchange)
```

Java

```
protected DoDataExchange(CDataExchange pDX);
```

MATLAB

```
DoDataExchange
```

Description

DDX/DDV support

Bind the form controls to member variables and set basic input range limits

Body Source

```
void CBodystatResultsDlg::DoDataExchange(CDataExchange* pDX)
{
 CDialog::DoDataExchange(pDX);

 DDX_Control(pDX, IDC_MODEL_COMBO, m_ModelComboCtrl);
 DDX_CBIndex(pDX, IDC_MODEL_COMBO, m_iModel);

 DDX_Radio(pDX, IDC_GENDER_MALE_RADIO, m_iGender);
 DDX_Text(pDX, IDC_AGE_EDIT, m_iAge);
 DDV_MinMaxInt(pDX, m_iAge, 1, 99);
 DDX_Text(pDX, IDC_HEIGHT_EDIT, m_iHeight);
 DDV_MinMaxInt(pDX, m_iHeight, 30, 229);
 DDX_Text(pDX, IDC_WEIGHT_EDIT, m_fWeight);
 DDV_MinMaxFloat(pDX, m_fWeight, 0.5, 300.0);
 DDX_CBIndex(pDX, IDC_ACTIVITY_COMBO, m_iActivityLevel);
 DDX_Text(pDX, IDC_WAIST_EDIT, m_iWaist);
 DDV_MinMaxInt(pDX, m_iWaist, 5, 648);
 DDX_Text(pDX, IDC_HIP_EDIT, m_iHip);
 DDV_MinMaxInt(pDX, m_iHip, 5, 648);
 DDX_Check(pDX, IDC_MODE_BODYCCOMP_CHECK, m_bModeBodyComposition);
 DDX_Check(pDX, IDC_MODE_HYDRATION_CHECK, m_bModeHydration);

 DDX_Text(pDX, IDC_IMP_5K_EDIT, m_iImpedance5K);
 DDV_MinMaxInt(pDX, m_iImpedance5K, 0, 2000);
 DDX_Text(pDX, IDC_IMP_50K_EDIT, m_iImpedance50K);
 DDV_MinMaxInt(pDX, m_iImpedance50K, 0, 2000);
 DDX_Text(pDX, IDC_IMP_100K_EDIT, m_iImpedance100K);
 DDV_MinMaxInt(pDX, m_iImpedance100K, 0, 2000);
 DDX_Text(pDX, IDC_IMP_200K_EDIT, m_iImpedance200K);
 DDV_MinMaxInt(pDX, m_iImpedance200K, 0, 2000);
 DDX_Text(pDX, IDC_RES_50K_EDIT, m_iResistance50K);
 DDV_MinMaxInt(pDX, m_iResistance50K, 0, 2000);
 DDX_Text(pDX, IDC_REAC_50K_EDIT, m_fReactance50K);
 DDV_MinMaxFloat(pDX, m_fReactance50K, 0, 200);
 DDX_Text(pDX, IDC_PHASEANGLE_50K_EDIT, m_fPhaseAngle50K);
 DDV_MinMaxFloat(pDX, m_fPhaseAngle50K, 0, 15);

 DDX_Control(pDX, IDC_RESULTS_LIST, m_ResultsListCtrl);
}
```

4.1.1.2.4.3 CBodystatResultsDlg::GetSystemErrorMessage Method

Gets a system error message.

C++

```
CString GetSystemErrorMessage(DWORD dwError);
```

C#

```
protected CString GetSystemErrorMessage(DWORD dwError);
```

Visual Basic

```
Protected Function GetSystemErrorMessage(dwError As DWORD) As CString
```

Java

```
protected CString GetSystemErrorMessage(DWORD dwError);
```

MATLAB

```
GetSystemErrorMessage
```

Parameters

Parameters	Description
DWORD dwError	The system error code to look up - typically from GetLastError()

Returns

The system error message as a string.

Body Source

```
CString CBodystatResultsDlg::GetSystemErrorMessage( DWORD dwError )
{
 LPTSTR  lpMsgBuf;

 FormatMessage(
  FORMAT_MESSAGE_ALLOCATE_BUFFER |
  FORMAT_MESSAGE_FROM_SYSTEM |
  FORMAT_MESSAGE_IGNORE_INSERTS,
  NULL,
  dwError,
  MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
  (LPTSTR) &lpMsgBuf,
  0, NULL );

 CString strErrorMsg;
 strErrorMsg = lpMsgBuf;

 return strErrorMsg;
}
```

4.1.1.2.4.4 CBodystatResultsDlg::OnClickedCalculateResultsButton Method

Executes the Calculate Results button action. Uses the Bodystat API to calculate the results and normals for this test measurement.

C++

```
afx_msg void OnClickedCalculateResultsButton();
```

C#

```
protected afx_msg void OnClickedCalculateResultsButton();
```

Visual Basic

```
Protected Function OnClickedCalculateResultsButton() As afx_msg void
```

Java

```
protected afx_msg void OnClickedCalculateResultsButton();
```

MATLAB

```
OnClickedCalculateResultsButton
```

Body Source

```
void CBodystatResultsDlg::OnClickedCalculateResultsButton()
{
 UpdateData(TRUE);      // Update member variables from form data

 m_ResultsListCtrl.DeleteAllItems();  // Clear the results list

 if ((BSDeviceModel)m_iModel == BSDeviceModel::BSQuadScanTouch || (BSDeviceModel)m_iModel
== BSDeviceModel::BSMultiScanTouch)
 {
  AfxMessageBox(_T("The selected model is not supported by this SDK sample."),
MB_ICONINFORMATION);
  return;
 }

 //
 // Prepare input structure for the Bodystat SDK (using member values from form data)
 // More likely this would be obtained directly from a BSMeasurement downloaded from the
unit
 //
 BSMeasurement bm;
 bm.iDeviceModel = (BSDeviceModel)m_iModel; // Use our combo selection for Bodystat model
type (its a direct mapping)
 bm.tTestDate = m_dtTestDate;    // Not relevant for performing calculations, but might be
copied to clipboard later
 bm.iGender = m_iGender == 0 ? Bodystat::BSMale : Bodystat::BSFemale;   // Convert our
gender radio buttons into Bodystat gender type
 bm.iAge = m_iAge;            // Age in years
 bm.iHeight = m_iHeight;         // Height in centimeters
 bm.fWeight = m_fWeight;         // Weight in kilograms
 bm.iActivity = m_iActivityLevel;   // Physical activity level (range 1 to 5) 1=very low
2=low/medium 3=medium 4=medium/high 5=very high
 bm.iWaist = m_iWaist;        // Waist in centimeters
 bm.iHip = m_iHip;            // Hip in centimeters
 bm.iZ_5kHz = m_iImpedance5K;      // Impedance 5 KHz  (in ohms) (MDD/Quad only)
 bm.iZ_50kHz = m_iImpedance50K;       // Impedance 50 KHz  (in ohms)
 bm.iZ_100kHz = m_iImpedance100K;   // Impedance 100 KHz (in ohms) (Quad only)
 bm.iZ_200kHz = m_iImpedance200K;   // Impedance 200 KHz (in ohms) (Quad only)
 bm.iR_50kHz = m_iResistance50K;    // Resistance 50 kHz (in ohms) (MDD/Quad only)
 bm.fX_50kHz = m_fReactance50K;    // Reactance 50 kHz  (in ohms) (MDD/Quad only)
 bm.fPA_50kHz = m_fPhaseAngle50K;   // Phase angle 50 kHz (in degrees) (MDD/Quad only)
 bm.iFrequencies = 0;      // Extended QST/MST frequencies not supported in this sample
 bm.pMultifreqData = NULL;     // Extended QST/MST frequencies not supported in this sample

 // Choose analysis mode based upon selected analysis checkboxes
 // These options might normally be stored as program setting rather than asked of the user
each time (like the installation mode of the device)
 // The analysis mode impacts the equations used behind some results in certain scenarios
 BSAnalysisMode iAnalysisMode;
 if(m_bModeBodyComposition && m_bModeHydration)
  iAnalysisMode = BSAnalysisModeBoth;
 else if(m_bModeBodyComposition)
  iAnalysisMode = BSAnalysisModeBC;
 else if(m_bModeHydration)
  iAnalysisMode = BSAnalysisModeHN;
 else
  iAnalysisMode = BSAnalysisModeNone;

 //
 // Get the results from the Bodystat SDK
 //
 BSError bseStatus;
 BSResults br;
 bseStatus = BSCalculateResults(&bm, &br, iAnalysisMode); // Results stored in br structure

 // Check for any errors from the calculation function
 if(bseStatus != Bodystat::NoError)
 {
  CString strMsg;
  strMsg.Format(_T("Error calculating results.\nError Code: %d"), (int)bseStatus);
  AfxMessageBox(strMsg, MB_ICONEXCLAMATION);

  return;  // Nothing to display
 }

 //
 // Get the normal ranges from the Bodystat SDK
 //
 BSNormals bn;
 bseStatus = BSCalculateNormals(&bm, &br, &bn);  // Normals stored in bn structure

 // Check for any errors from the normals function
 if(bseStatus != Bodystat::NoError)
 {
  CString strMsg;
  strMsg.Format(_T("Error calculating normals.\nError Code: %d"), (int)bseStatus);
  AfxMessageBox(strMsg, MB_ICONEXCLAMATION);
```

```
    return;  // Nothing to display
   }

   // Populate the list control with the calculated results and calculated normals
   PopulateListCtrl(bm, br, bn);

  }
```

4.1.1.2.4.5 CBodystatResultsDlg::OnClickedCopyToClipboardButton Method

Executes the Copy To Clipboard button action.

C++

```
afx_msg void OnClickedCopyToClipboardButton();
```

C#

```
protected afx_msg void OnClickedCopyToClipboardButton();
```

Visual Basic

```
Protected Function OnClickedCopyToClipboardButton() As afx_msg void
```

Java

```
protected afx_msg void OnClickedCopyToClipboardButton();
```

MATLAB

```
OnClickedCopyToClipboardButton
```

Body Source

```
void CBodystatResultsDlg::OnClickedCopyToClipboardButton()
{
 CopyToClipboard();
}
```

4.1.1.2.4.6 CBodystatResultsDlg::OnInitDialog Method

C++

```
virtual BOOL OnInitDialog();
```

C#

```
public BOOL OnInitDialog();
```

Visual Basic

```
Public Function OnInitDialog() As BOOL
```

Java

```
public BOOL OnInitDialog();
```

MATLAB

```
OnInitDialog
```

Description

CBodystatResultsDlg (see page 185) message handlers

Body Source

```
BOOL CBodystatResultsDlg::OnInitDialog()
{
 CDialog::OnInitDialog();

 // Setup the results list control with three columns ready for the results and normal
values
 m_ResultsListCtrl.InsertColumn(0, _T("Result"), 0, 250);
 m_ResultsListCtrl.InsertColumn(1, _T("Value"), 0, 110);
 m_ResultsListCtrl.InsertColumn(2, _T("Normal"), 0, 140);
 m_ResultsListCtrl.SetExtendedStyle( m_ResultsListCtrl.GetExtendedStyle() |
LVS_EX_FULLROWSELECT );
```

```
  //
  // Populate model combo with all possible models
  //
  int nItem = m_ModelComboCtrl.InsertString(0, _T("Unknown"));
  CString strModel, strComboItem;
  BSError bsErr = NoError;
  int iModel = 1;
  do
  {
   bsErr = BSGetDeviceModelName((BSDeviceModel)iModel, strModel.GetBuffer(32), 32);
   strModel.ReleaseBuffer();
   if(bsErr == NoError)
   {
    strComboItem.Format(_T("%s (ID:%d)"), strModel, iModel);  // Append model id number to
the string (for illustration and to make it unique for the combo)
    nItem = m_ModelComboCtrl.InsertString(iModel, strComboItem);
   }
   iModel++;
  }while(bsErr == NoError);   // Stop when we run out of valid model ids

  // Select the desired model
  m_ModelComboCtrl.SetCurSel(m_iModel);

  return TRUE;  // return TRUE unless you set the focus to a control
  // EXCEPTION: OCX Property Pages should return FALSE
}
```

4.1.1.2.4.7 CBodystatResultsDlg::PopulateListCtrl Method

Populate the list control with results and normals.

C++

```
void PopulateListCtrl(const BSMeasurement & bm, const BSResults & br, const BSNormals & bn);
```

C#

```
protected PopulateListCtrl(BSMeasurement & bm, BSResults & br, BSNormals & bn);
```

Visual Basic

```
Protected Sub PopulateListCtrl(bm As BSMeasurement &, br As BSResults &, bn As BSNormals &)
```

Java

```
protected PopulateListCtrl(final BSMeasurement & bm, final BSResults & br, final BSNormals
& bn);
```

MATLAB

```
PopulateListCtrl
```

Parameters

Parameters	Description
const BSMeasurement & bm	The measurement parameters for this test.
const BSResults & br	The calculated results for this test from the SDK.
const BSNormals & bn	The calculated normals for this test from the SDK.

Body Source

```
void CBodystatResultsDlg::PopulateListCtrl( const BSMeasurement &bm, const BSResults &br,
const BSNormals &bn )
{
 CString strResult, strNormal;
 int nItem;

 // Fat %
 nItem = m_ResultsListCtrl.InsertItem(0, _T("Fat percentage"));
 strResult.Format(_T("%.1f"), br.fFatPerc);  m_ResultsListCtrl.SetItemText(nItem, 1,
strResult);
 strNormal.Format(_T("%d - %d"), bn.iFatPerc_L, bn.iFatPerc_H);
m_ResultsListCtrl.SetItemText(nItem, 2, strNormal);
```

```
 // Fat kg
 nItem = m_ResultsListCtrl.InsertItem(1, _T("Fat (in kg)"));
 strResult.Format(_T("%.1f"), br.fFatKg);  m_ResultsListCtrl.SetItemText(nItem, 1,
strResult);
 strNormal.Format(_T("%d - %d"), bn.iFatKg_L, bn.iFatKg_H);
m_ResultsListCtrl.SetItemText(nItem, 2, strNormal);

 // Lean %
 nItem = m_ResultsListCtrl.InsertItem(2, _T("Lean percentage"));
 strResult.Format(_T("%.1f"), br.fLeanPerc);  m_ResultsListCtrl.SetItemText(nItem, 1,
strResult);
 strNormal.Format(_T("%d - %d"), bn.iLeanPerc_L, bn.iLeanPerc_H);
m_ResultsListCtrl.SetItemText(nItem, 2, strNormal);

 // Lean kg
 nItem = m_ResultsListCtrl.InsertItem(3, _T("Lean (in kg)"));
 strResult.Format(_T("%.1f"), br.fLeanKg);  m_ResultsListCtrl.SetItemText(nItem, 1,
strResult);
 strNormal.Format(_T("%d - %d"), bn.iLeanKg_L, bn.iLeanKg_H);
m_ResultsListCtrl.SetItemText(nItem, 2, strNormal);

 // Total Weight
 nItem = m_ResultsListCtrl.InsertItem(4, _T("Total Weight (in kg)"));
 strResult.Format(_T("%.1f"), br.fTotalWeight); m_ResultsListCtrl.SetItemText(nItem, 1,
strResult);
 // Total weight normal range is calculated using composition or BMI (whichever is the
greater). We indicate which method was used in brackets after the value.
 strNormal.Format(_T("%d - %d (%s)"), bn.iTotalWeight_L, bn.iTotalWeight_H,
bn.iTotalWeightMethod == 0 ? _T("Composition") : _T("BMI"));
m_ResultsListCtrl.SetItemText(nItem, 2, strNormal);

 // Dry lean weight
 nItem = m_ResultsListCtrl.InsertItem(5, _T("Dry Lean Weight (in kg)"));
 strResult.Format(_T("%.1f"), br.fDryLW);  m_ResultsListCtrl.SetItemText(nItem, 1,
strResult);
 strNormal.Format(_T("N/A"));  m_ResultsListCtrl.SetItemText(nItem, 2, strNormal);

 // TBW %
 nItem = m_ResultsListCtrl.InsertItem(6, _T("Total Body Water percentage"));
 strResult.Format(_T("%.1f"), br.fTBWPerc);  m_ResultsListCtrl.SetItemText(nItem, 1,
strResult);
 strNormal.Format(_T("%d - %d"), bn.iTBWPerc_L, bn.iTBWPerc_H);
m_ResultsListCtrl.SetItemText(nItem, 2, strNormal);

 // TBW lt
 nItem = m_ResultsListCtrl.InsertItem(7, _T("Total Body Water (in litres)"));
 strResult.Format(_T("%.1f"), br.fTBW);   m_ResultsListCtrl.SetItemText(nItem, 1,
strResult);
 strNormal.Format(_T("%d - %d"), bn.iTBW_L, bn.iTBW_H);
m_ResultsListCtrl.SetItemText(nItem, 2, strNormal);

 // ECW %
 nItem = m_ResultsListCtrl.InsertItem(8, _T("Extra Cellular Water percentage"));
 BSGetDeviceFamily(bm.iDeviceModel) == BSQuadScan_Family ||
BSGetDeviceFamily(bm.iDeviceModel) == BSMDD_Family ? strResult.Format(_T("%.1f"),
br.fECWPerc) : strResult = _T("(MDD/Quad Only)");  m_ResultsListCtrl.SetItemText(nItem, 1,
strResult);
 BSGetDeviceFamily(bm.iDeviceModel) == BSQuadScan_Family ||
BSGetDeviceFamily(bm.iDeviceModel) == BSMDD_Family ? strNormal.Format(_T("%d"),
bn.iECWPerc_Norm) : strNormal = _T("(MDD/Quad Only)");
m_ResultsListCtrl.SetItemText(nItem, 2, strNormal);

 // ECW lt
 nItem = m_ResultsListCtrl.InsertItem(9, _T("Extra Cellular Water (in litres)"));
 BSGetDeviceFamily(bm.iDeviceModel) == BSQuadScan_Family ||
BSGetDeviceFamily(bm.iDeviceModel) == BSMDD_Family ? strResult.Format(_T("%.1f"), br.fECW)
: strResult = _T("(MDD/Quad Only)");   m_ResultsListCtrl.SetItemText(nItem, 1, strResult);
 strNormal.Format(_T("N/A"));  m_ResultsListCtrl.SetItemText(nItem, 2, strNormal);

 // ICW %
 nItem = m_ResultsListCtrl.InsertItem(10, _T("Intra Cellular Water percentage"));
 BSGetDeviceFamily(bm.iDeviceModel) == BSQuadScan_Family ||
BSGetDeviceFamily(bm.iDeviceModel) == BSMDD_Family ? strResult.Format(_T("%.1f"),
br.fICWPerc) : strResult = _T("(MDD/Quad Only)");  m_ResultsListCtrl.SetItemText(nItem, 1,
strResult);
 BSGetDeviceFamily(bm.iDeviceModel) == BSQuadScan_Family ||
BSGetDeviceFamily(bm.iDeviceModel) == BSMDD_Family ? strNormal.Format(_T("%d"),
bn.iICWPerc_Norm) : strNormal = _T("(MDD/Quad Only)");
m_ResultsListCtrl.SetItemText(nItem, 2, strNormal);

 // ICW lt
 nItem = m_ResultsListCtrl.InsertItem(11, _T("Intra Cellular Water (in litres)"));
 BSGetDeviceFamily(bm.iDeviceModel) == BSQuadScan_Family ||
BSGetDeviceFamily(bm.iDeviceModel) == BSMDD_Family ? strResult.Format(_T("%.1f"), br.fICW)
: strResult = _T("(MDD/Quad Only)");   m_ResultsListCtrl.SetItemText(nItem, 1, strResult);
 strNormal.Format(_T("N/A"));  m_ResultsListCtrl.SetItemText(nItem, 2, strNormal);

 // Body Cell Mass
 nItem = m_ResultsListCtrl.InsertItem(12, _T("Body Cell Mass"));
 BSGetDeviceFamily(bm.iDeviceModel) == BSQuadScan_Family ? strResult.Format(_T("%.1f"),
br.fBCM) : strResult = _T("(Quad Only)");   m_ResultsListCtrl.SetItemText(nItem, 1,
strResult);
 strNormal.Format(_T("N/A"));  m_ResultsListCtrl.SetItemText(nItem, 2, strNormal);

 // 3rd space water
 nItem = m_ResultsListCtrl.InsertItem(13, _T("3rd space water (in litres)"));
 BSGetDeviceFamily(bm.iDeviceModel) == BSQuadScan_Family ? strResult.Format(_T("%.1f"),
br.fThirdSpace) : strResult = _T("(Quad Only)"); m_ResultsListCtrl.SetItemText(nItem, 1,
strResult);
 strNormal.Format(_T("N/A"));  m_ResultsListCtrl.SetItemText(nItem, 2, strNormal);

 // Nutrition index
 nItem = m_ResultsListCtrl.InsertItem(14, _T("Nutrition index"));
 BSGetDeviceFamily(bm.iDeviceModel) == BSQuadScan_Family ? strResult.Format(_T("%.2f"),
br.fNutrition) : strResult = _T("(Quad Only)"); m_ResultsListCtrl.SetItemText(nItem, 1,
strResult);
 BSGetDeviceFamily(bm.iDeviceModel) == BSQuadScan_Family ? strNormal.Format(_T("%.2f"),
bn.fNutrition_Norm) : strNormal = _T("(Quad Only)");  m_ResultsListCtrl.SetItemText(nItem,
2, strNormal);

 // Prediction Marker (TM) - formerly known as Illness Marker (TM)
 nItem = m_ResultsListCtrl.InsertItem(15, _T("Prediction Marker (TM)"));
 BSGetDeviceFamily(bm.iDeviceModel) == BSQuadScan_Family ? strResult.Format(_T("%.3f"),
br.fIllness) : strResult = _T("(Quad Only)");  m_ResultsListCtrl.SetItemText(nItem, 1,
strResult);
 BSGetDeviceFamily(bm.iDeviceModel) == BSQuadScan_Family ? strNormal.Format(_T("%.3f -
%.3f"), bn.fIllness_L, bn.fIllness_H) : strNormal = _T("(Quad Only)");
m_ResultsListCtrl.SetItemText(nItem, 2, strNormal);

 // BMR kcal
 nItem = m_ResultsListCtrl.InsertItem(16, _T("Basal Metabolic Rate (in kcal)"));
 strResult.Format(_T("%.0f"), br.fBMR);   m_ResultsListCtrl.SetItemText(nItem, 1,
strResult);
 strNormal.Format(_T("N/A"));  m_ResultsListCtrl.SetItemText(nItem, 2, strNormal);

 // BMR kcal/kg
 nItem = m_ResultsListCtrl.InsertItem(17, _T("Basal Metabolic Rate per kilogram (in
kcal/kg)"));
 strResult.Format(_T("%.1f"), br.fBMRkg);  m_ResultsListCtrl.SetItemText(nItem, 1,
strResult);
 strNormal.Format(_T("N/A"));  m_ResultsListCtrl.SetItemText(nItem, 2, strNormal);

 // EAR
 nItem = m_ResultsListCtrl.InsertItem(18, _T("Estimated Average Requirement (in kcal)"));
 strResult.Format(_T("%.0f"), br.fEstAvg);  m_ResultsListCtrl.SetItemText(nItem, 1,
strResult);
 strNormal.Format(_T("N/A"));  m_ResultsListCtrl.SetItemText(nItem, 2, strNormal);

 // BMI
 nItem = m_ResultsListCtrl.InsertItem(19, _T("Body Mass Index (BMI)"));
 strResult.Format(_T("%.1f"), br.fBMI);   m_ResultsListCtrl.SetItemText(nItem, 1,
strResult);
 strNormal.Format(_T("%d - %d"), bn.iBMI_L, bn.iBMI_H);
m_ResultsListCtrl.SetItemText(nItem, 2, strNormal);
```

```
 // BFMI
 nItem = m_ResultsListCtrl.InsertItem(20, _T("Body Fat Mass Index (BFMI)"));
 BSGetDeviceFamily(bm.iDeviceModel) == BSQuadScan_Family ||
BSGetDeviceFamily(bm.iDeviceModel) == BSMDD_Family ? strResult.Format(_T("%.1f"), br.fBFMI)
: strResult = _T("(MDD/Quad Only)");  m_ResultsListCtrl.SetItemText(nItem, 1, strResult);
 BSGetDeviceFamily(bm.iDeviceModel) == BSQuadScan_Family ||
BSGetDeviceFamily(bm.iDeviceModel) == BSMDD_Family ? strNormal.Format(_T("%d - %d"),
bn.iBFMI_L, bn.iBFMI_H) : strNormal = _T("(MDD/Quad Only)");
m_ResultsListCtrl.SetItemText(nItem, 2, strNormal);

 // FFMI
 nItem = m_ResultsListCtrl.InsertItem(21, _T("Fat Free Mass Index (FFMI)"));
 BSGetDeviceFamily(bm.iDeviceModel) == BSQuadScan_Family ||
BSGetDeviceFamily(bm.iDeviceModel) == BSMDD_Family ? strResult.Format(_T("%.1f"), br.fFFMI)
: strResult = _T("(MDD/Quad Only)");  m_ResultsListCtrl.SetItemText(nItem, 1, strResult);
 BSGetDeviceFamily(bm.iDeviceModel) == BSQuadScan_Family ||
BSGetDeviceFamily(bm.iDeviceModel) == BSMDD_Family ? strNormal.Format(_T("%d - %d"),
bn.iFFMI_L, bn.iFFMI_H) : strNormal = _T("(MDD/Quad Only)");
m_ResultsListCtrl.SetItemText(nItem, 2, strNormal);

 // Waist/Hip
 nItem = m_ResultsListCtrl.InsertItem(22, _T("Waist/Hip ratio"));
 strResult.Format(_T("%.2f"), br.fWaistHip);  m_ResultsListCtrl.SetItemText(nItem, 1,
strResult);
 strNormal.Format(_T("%.2f"), bn.fWaistHip_Norm);  m_ResultsListCtrl.SetItemText(nItem, 2,
strNormal);

 // Wellness marker
 nItem = m_ResultsListCtrl.InsertItem(23, _T("Wellness Marker (TM)"));
 BSGetDeviceFamily(bm.iDeviceModel) == BSMDD_Family ? strResult.Format(_T("%.3f"),
br.fWellness) : strResult = _T("(MDD Only)");   m_ResultsListCtrl.SetItemText(nItem, 1,
strResult);
 BSGetDeviceFamily(bm.iDeviceModel) == BSMDD_Family ? strNormal.Format(_T("%.3f - %.3f"),
bn.fWellness_L, bn.fWellness_H) : strNormal = _T("(MDD Only)");
m_ResultsListCtrl.SetItemText(nItem, 2, strNormal);
}
```

4.2 Bodystat C# WinForms Sample

A sample application written in C# .NET based upon WinForms within the Microsoft .NET Framework. Solutions are provided for Visual Studio 2008 and 2010. You must have Visual Studiio 2008 or 2010 installed to work with this sample.

Namespaces

Name	Description
BodystatCSWinForms (see page 205)	This is the main namespace for our sample project.

Files

Name	Description
MainForm.cs (see page 216)	Bodystat VB.Net Sample (C) 2010-2012 Bodystat (Isle of Man) Ltd. All rights reserved. Use subject to license. Web: http://www.bodystat.com Email: info@bodystat.com Tel: +44 (0)1624 629 571 Fax: +44 (0)1624 611 544

ResultsForm.cs (see page 216)	Bodystat C# WinForms Sample (C) 2010-2014 Bodystat (Isle of Man) Ltd. All rights reserved. Use subject to license. Web: http://www.bodystat.com Email: info@bodystat.com Tel: +44 (0)1624 629 571 Fax: +44 (0)1624 611 544

4.2.1 BodystatCSWinForms Namespace

This is the main namespace for our sample project.

Module

Bodystat C# WinForms Sample (see page 204)

Classes

	Name	Description
	MainForm (see page 205)	Main sample dialog. Provides buttons utilising the various functions of the Bodystat API. A list control also shows data (test measurements) downloaded from the device.
	ResultsForm (see page 209)	Results sample dialog. Displays the various parameters relating to a measurement (allowing direct input by the user), then utilises the Bodystat API to calculate detailed results and normals.

4.2.1.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	MainForm (see page 205)	Main sample dialog. Provides buttons utilising the various functions of the Bodystat API. A list control also shows data (test measurements) downloaded from the device.
	ResultsForm (see page 209)	Results sample dialog. Displays the various parameters relating to a measurement (allowing direct input by the user), then utilises the Bodystat API to calculate detailed results and normals.

4.2.1.1.1 MainForm Class

Main sample dialog. Provides buttons utilising the various functions of the Bodystat API. A list control also shows data (test measurements) downloaded from the device.

Class Hierarchy

Form
BodystatCSWinForms.MainForm

C++

```
class MainForm : public Form;
```

C#

```
public class MainForm : Form;
```

Visual Basic

```
Public Class MainForm
```

```
Inherits Form
```

Java

```
public class MainForm Form;
```

MATLAB

```
MainForm
```

File

MainForm.cs (see page 216)

Methods

	Name	Description
◆	MainForm (see page 206)	Prepare the controls on the sample dialog

MainForm Methods

	Name	Description
◆	PopulateMeasurementList (see page 207)	Populate the measurement list control with test measurement data from the API. Illustrates iteration through a BSRawData structure, working with data in a BSMeasurement structure and helper functions from the Bodystat API like BSGetDeviceModelName

4.2.1.1.1.1 MainForm.MainForm Constructor

Prepare the controls on the sample dialog

C++

```
MainForm();
```

C#

```
public MainForm();
```

Visual Basic

```
Public Sub New()
```

Java

```
public MainForm();
```

MATLAB

```
MainForm
```

Body Source

```
public MainForm()
     {
         InitializeComponent();

         // Add columns to the test measurement list ready for download of data
this.lstviewTestMeasurementList.Columns.Add("Test No", 50);
this.lstviewTestMeasurementList.Columns.Add("Test Date/Time", 130);
this.lstviewTestMeasurementList.Columns.Add("Model", 140);
this.lstviewTestMeasurementList.Columns.Add("Serial No", 75);
this.lstviewTestMeasurementList.Columns.Add("Gender", 75);
this.lstviewTestMeasurementList.Columns.Add("Age", 75);
this.lstviewTestMeasurementList.Columns.Add("Height", 75);
this.lstviewTestMeasurementList.Columns.Add("Weight", 75);
this.lstviewTestMeasurementList.Columns.Add("Activity", 75);
this.lstviewTestMeasurementList.Columns.Add("Waist", 75);
this.lstviewTestMeasurementList.Columns.Add("Hip", 75);
this.lstviewTestMeasurementList.Columns.Add("Imp 5 kHz", 75);
this.lstviewTestMeasurementList.Columns.Add("Imp 50 kHz", 75);
this.lstviewTestMeasurementList.Columns.Add("Imp 100 kHz", 75);
this.lstviewTestMeasurementList.Columns.Add("Imp 200 kHz", 75);
this.lstviewTestMeasurementList.Columns.Add("Res 50 kHz", 75);
this.lstviewTestMeasurementList.Columns.Add("Reac 50 kHz", 75);
this.lstviewTestMeasurementList.Columns.Add("Phase 50 kHz", 80);
```

```
       }
```

4.2.1.1.1.2 MainForm Methods

4.2.1.1.1.2.1 MainForm.PopulateMeasurementList Method

Populate the measurement list control with test measurement data from the API. Illustrates iteration through a BSRawData structure, working with data in a BSMeasurement structure and helper functions from the Bodystat API like BSGetDeviceModelName

C++

```
void PopulateMeasurementList(BodystatAPI.BodystatAPI.BSRawData ^ pRawData);
```

C#

```
public void PopulateMeasurementList(ref BodystatAPI.BodystatAPI.BSRawData pRawData);
```

Visual Basic

```
Public Function PopulateMeasurementList(ByRef pRawData As
BodystatAPI.BodystatAPI.BSRawData) As void
```

Java

```
public void PopulateMeasurementList(BodystatAPI.BodystatAPI.BSRawData pRawData);
```

MATLAB

```
PopulateMeasurementList
```

Parameters

Parameters	Description
ref BodystatAPI.BodystatAPI.BSRawData pRawData	[in] RawData to be parsed.

Body Source

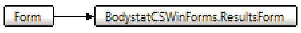
```
public void PopulateMeasurementList(ref BodystatAPI.BodystatAPI.BSRawData pRawData)
{
 // Clear any previous items in the list
 this.lstviewTestMeasurementList.Items.Clear();

 string strItem = null;

 System.Windows.Forms.ListViewItem lvi = null;
 int i = 0;

 // Process each record in the structure

 for (i = 0; i <= pRawData.iTotalNumRecs - 1; i++) {
  // Test no
  strItem = string.Format("{0}", i + pRawData.ulFirstTestNum);
  lvi = lstviewTestMeasurementList.Items.Insert(i, strItem);
  lvi.Tag = pRawData.record[i];
  // Assign this BSMeasurement to the listview item (so we can reference it later)

  // Test date/time
  lvi.SubItems.Add(pRawData.record[i].tTestDate.ToString());

  // Device model
        BodystatAPI.BodystatAPI.BSGetDeviceModelName(pRawData.record[i].iDeviceModel, out
strItem);
              lvi.SubItems.Add(strItem);

  // Device serial number
  strItem = string.Format("{0}", pRawData.record[i].ulDeviceSerialNumber);
  lvi.SubItems.Add(strItem);

  // Gender
  lvi.SubItems.Add((pRawData.record[i].iGender == BSGender.BSMale ? "Male" : "Female"));

  // Age
```

```
    strItem = string.Format("{0}", pRawData.record[i].iAge);
    lvi.SubItems.Add(strItem);

    // Height
    strItem = string.Format("{0} cm", pRawData.record[i].iHeight);
    lvi.SubItems.Add(strItem);

    // Weight
    strItem = string.Format("{0:F1} kg", pRawData.record[i].fWeight);
    lvi.SubItems.Add(strItem);

    // Activity
    switch (pRawData.record[i].iActivity) {
     case 1:
      strItem = "Very Low";
      break;
     case 2:
      strItem = "Low/Medium";
      break;
     case 3:
      strItem = "Medium";
      break;
     case 4:
      strItem = "Medium/High";
      break;
     case 5:
      strItem = "Very High";
      break;
     default:
      strItem = "Unknown";
      break;
    }
    lvi.SubItems.Add(strItem);

    // Waist
    strItem = string.Format("{0} cm", pRawData.record[i].iWaist);
    lvi.SubItems.Add(strItem);

    // Hip
    strItem = string.Format("{0} cm", pRawData.record[i].iHip);
    lvi.SubItems.Add(strItem);

    // Z5 (Impedance at 5 kHz)
    strItem = string.Format("{0} ohm", pRawData.record[i].iZ_5kHz);
    lvi.SubItems.Add(strItem);

    // Z50 (Impedance at 50 kHz)
    strItem = string.Format("{0} ohm", pRawData.record[i].iZ_50kHz);
    lvi.SubItems.Add(strItem);

    // Z100 (Impedance at 100 kHz)
    strItem = string.Format("{0} ohm", pRawData.record[i].iZ_100kHz);
    lvi.SubItems.Add(strItem);

    // Z200 (Impedance at 200 kHz)
    strItem = string.Format("{0} ohm", pRawData.record[i].iZ_200kHz);
    lvi.SubItems.Add(strItem);

    // R50 (Resistance at 50 kHz)
    strItem = string.Format("{0} ohm", pRawData.record[i].iR_50kHz);
    lvi.SubItems.Add(strItem);

    // X50 (Reactance at 50 kHz)
    strItem = string.Format("{0:F1} ohm", pRawData.record[i].fX_50kHz);
    lvi.SubItems.Add(strItem);

    // PA50 (Phase Angle at 50 kHz)
    strItem = string.Format("{0:F1} deg", pRawData.record[i].fPA_50kHz);
    lvi.SubItems.Add(strItem);

   }
  }
```

4.2.1.1.2 ResultsForm Class

Results sample dialog. Displays the various parameters relating to a measurement (allowing direct input by the user), then utilises the Bodystat API to calculate detailed results and normals.

Class Hierarchy



C++

```
class ResultsForm : public Form;
```

C#

```
public class ResultsForm : Form;
```

Visual Basic

```
Public Class ResultsForm
Inherits Form
```

Java

```
public class ResultsForm Form;
```

MATLAB

```
ResultsForm
```

File

ResultsForm.cs (see page 216)

Methods

	Name	Description
◆	ResultsForm (see page 209)	Initialise the dialog controls with their content

ResultsForm Methods

	Name	Description
◆	InitialiseFromMeasurement (see page 210)	Initialise the dialog controls with a values from an actual measurement result
◆	InitialiseSampleMeasurement (see page 211)	Initialise the dialog controls with a sample measurement result
◆	PopulateListCtrl (see page 212)	Populate the list control with results and normals.
◆	UseBSMeasurement (see page 215)	Allows the caller to specify a downloaded measurement structure to be use when initialising the dialog. Must be called before the form is loaded (displayed)

4.2.1.1.2.1 ResultsForm.ResultsForm Constructor

Initialise the dialog controls with their content

C++

```
ResultsForm();
```

C#

```
public ResultsForm();
```

Visual Basic

```
Public Sub New()
```

Java

```
public ResultsForm();
```

MATLAB

```
ResultsForm
```

Body Source

```
public ResultsForm()
{
    InitializeComponent();

    //
    // Populate model combo with all possible models
    //
    this.cboModelCombo.Items.Add("Unknown");
    string strModel;
    string strComboItem = null;
    BodystatAPI.BSError bsErr = BodystatAPI.BSError.NoError;
    int iModel = 1;
    do
    {
        bsErr = BodystatAPI.BodystatAPI.BSGetDeviceModelName((BSDeviceModel)iModel, out
strModel);

        if ((bsErr == BodystatAPI.BSError.NoError))
        {
            strComboItem = string.Format("{0} (ID:{1})", strModel, iModel);
            // Append model id number to the string (for illustration and to make it unique
for the combo)
            this.cboModelCombo.Items.Add(strComboItem);

        }
        iModel += 1;
    } while ((bsErr == BodystatAPI.BSError.NoError));
    // Stop when we run out of valid model ids

    // Setup the results list control with three columns ready for the results and normal
values
    lstviewResultsList.Columns.Add("Result", 250);
    lstviewResultsList.Columns.Add("Value", 110);
    lstviewResultsList.Columns.Add("Normal", 140);
}
```

4.2.1.1.2.2 ResultsForm Methods

4.2.1.1.2.2.1 ResultsForm.InitialiseFromMeasurement Method

Initialise the dialog controls with a values from an actual measurement result

C++

```
void InitialiseFromMeasurement();
```

C#

```
public void InitialiseFromMeasurement();
```

Visual Basic

```
Public Function InitialiseFromMeasurement() As void
```

Java

```
public void InitialiseFromMeasurement();
```

MATLAB

```
InitialiseFromMeasurement
```

Body Source

```
public void InitialiseFromMeasurement()
{
    this.cboModelCombo.SelectedIndex = (int)bsmMeasurement.iDeviceModel;
    //Me.dtTestDate = bsmMeasurement.tTestDate       ' Not shown on dialog presently
    this.rdoGenderMaleRadio.Checked = bsmMeasurement.iGender == BodystatAPI.BSGender.BSMale;
    this.rdoGenderFemaleRadio.Checked = bsmMeasurement.iGender ==
BodystatAPI.BSGender.BSFemale;
    this.txtAgeEdit.Text = bsmMeasurement.iAge.ToString();
    this.txtHeightEdit.Text = bsmMeasurement.iHeight.ToString();
    this.txtWeightEdit.Text = bsmMeasurement.fWeight.ToString();
    this.cboActivityCombo.SelectedIndex = bsmMeasurement.iActivity;
    this.txtWaistEdit.Text = bsmMeasurement.iWaist.ToString();
    this.txtHipEdit.Text = bsmMeasurement.iHip.ToString();

    this.chkModeBodyCompCheck.Checked = true;
    this.chkModeHydrationCheck.Checked = true;

    this.txtImp5kEdit.Text = bsmMeasurement.iZ_5kHz.ToString();
    this.txtImp50kEdit.Text = bsmMeasurement.iZ_50kHz.ToString();
    this.txtImp100kEdit.Text = bsmMeasurement.iZ_100kHz.ToString();
    this.txtImp200kEdit.Text = bsmMeasurement.iZ_200kHz.ToString();
    this.txtRes50kEdit.Text = bsmMeasurement.iR_50kHz.ToString();
    this.txtReac50kEdit.Text = bsmMeasurement.fX_50kHz.ToString();
    this.txtPhaseAngle50kEdit.Text = bsmMeasurement.fPA_50kHz.ToString();

}
```

4.2.1.1.2.2.2 ResultsForm.InitialiseSampleMeasurement Method

Initialise the dialog controls with a sample measurement result

C++

```
void InitialiseSampleMeasurement();
```

C#

```
public void InitialiseSampleMeasurement();
```

Visual Basic

```
Public Function InitialiseSampleMeasurement() As void
```

Java

```
public void InitialiseSampleMeasurement();
```

MATLAB

```
InitialiseSampleMeasurement
```

Body Source

```
public void InitialiseSampleMeasurement()
{
    // Setup the form with some sensible default values
    this.cboModelCombo.SelectedIndex = (int)BodystatAPI.BSDeviceModel.BSQuadScan_BT;
    //Me.dtTestDate = DateTime.Now()       ' Not shown on dialog presently
    this.rdoGenderMaleRadio.Checked = true;
    this.rdoGenderFemaleRadio.Checked = false;
    this.txtAgeEdit.Text = "30";
    this.txtHeightEdit.Text = "165";
    this.txtWeightEdit.Text = "85.0";
    this.cboActivityCombo.SelectedIndex = 3;
    this.txtWaistEdit.Text = "85";
    this.txtHipEdit.Text = "90";

    this.chkModeBodyCompCheck.Checked = true;
    this.chkModeHydrationCheck.Checked = true;

    this.txtImp5kEdit.Text = "500";
    this.txtImp50kEdit.Text = "500";
    this.txtImp100kEdit.Text = "500";
    this.txtImp200kEdit.Text = "500";
    this.txtRes50kEdit.Text = "500";
    this.txtReac50kEdit.Text = "52.0";
    this.txtPhaseAngle50kEdit.Text = "6.1";
}
```

4.2.1.1.2.2.3 ResultsForm.PopulateListCtrl Method

Populate the list control with results and normals.

C++

```
void PopulateListCtrl(BodystatAPI.BodystatAPI.BSMeasurement ^ bm,
BodystatAPI.BodystatAPI.BSResults ^ br, BodystatAPI.BodystatAPI.BSNormals ^ bn);
```

C#

```
public void PopulateListCtrl(ref BodystatAPI.BodystatAPI.BSMeasurement bm, ref
BodystatAPI.BodystatAPI.BSResults br, ref BodystatAPI.BodystatAPI.BSNormals bn);
```

Visual Basic

```
Public Function PopulateListCtrl(ByRef bm As BodystatAPI.BodystatAPI.BSMeasurement, ByRef
br As BodystatAPI.BodystatAPI.BSResults, ByRef bn As BodystatAPI.BodystatAPI.BSNormals) As
void
```

Java

```
public void PopulateListCtrl(BodystatAPI.BodystatAPI.BSMeasurement bm,
BodystatAPI.BodystatAPI.BSResults br, BodystatAPI.BodystatAPI.BSNormals bn);
```

MATLAB

```
PopulateListCtrl
```

Parameters

Parameters	Description
ref BodystatAPI.BodystatAPI.BSMeasurement bm	The measurement parameters for this test.
ref BodystatAPI.BodystatAPI.BSResults br	The calculated results for this test from the SDK.
ref BodystatAPI.BodystatAPI.BSNormals bn	The calculated normals for this test from the SDK.

Body Source

```
public void PopulateListCtrl(ref BodystatAPI.BodystatAPI.BSMeasurement bm, ref
BodystatAPI.BodystatAPI.BSResults br, ref BodystatAPI.BodystatAPI.BSNormals bn)
{
    ListViewItem lvi = default(ListViewItem);

    // Fat %
    lvi = lstviewResultsList.Items.Add("Fat percentage");
    lvi.SubItems.Add(string.Format("{0:F1}", br.FatPerc));
    lvi.SubItems.Add(string.Format("{0} - {1}", bn.FatPerc_L, bn.FatPerc_H));

    // Fat kg
    lvi = lstviewResultsList.Items.Add("Fat (in kg)");
    lvi.SubItems.Add(string.Format("{0:F1}", br.FatKg));
    lvi.SubItems.Add(string.Format("{0} - {1}", bn.FatKg_L, bn.FatKg_H));

    // Lean %
    lvi = lstviewResultsList.Items.Add("Lean percentage");
    lvi.SubItems.Add(string.Format("{0:F1}", br.LeanPerc));
    lvi.SubItems.Add(string.Format("{0} - {1}", bn.LeanPerc_L, bn.LeanPerc_H));

    // Lean kg
    lvi = lstviewResultsList.Items.Add("Lean (in kg)");
    lvi.SubItems.Add(string.Format("{0:F1}", br.LeanKg));
    lvi.SubItems.Add(string.Format("{0} - {1}", bn.LeanKg_L, bn.LeanKg_H));

    // Total Weight
    lvi = lstviewResultsList.Items.Add("Total Weight (in kg)");
    lvi.SubItems.Add(string.Format("{0:F1}", br.TotalWeight));
    // Total weight normal range is calculated using composition or BMI (whichever is the
greater). We indicate which method was used in brackets after the value.
    lvi.SubItems.Add(string.Format("{0} - {1} ({2})", bn.TotalWeight_L, bn.TotalWeight_H,
(bn.TotalWeightMethod == 0 ? "Composition" : "BMI")));

    // Dry lean weight
    lvi = lstviewResultsList.Items.Add("Dry Lean Weight (in kg)");
```

```
        lvi.SubItems.Add(string.Format("{0:F1}", br.DryLW));
        lvi.SubItems.Add("N/A");

        // TBW %
        lvi = lstviewResultsList.Items.Add("Total Body Water percentage");
        lvi.SubItems.Add(string.Format("{0:F1}", br.TBWPerc));
        lvi.SubItems.Add(string.Format("{0} - {1}", bn.TBWPerc_L, bn.TBWPerc_H));

        // TBW lt
        lvi = lstviewResultsList.Items.Add("Total Body Water (in litres)");
        lvi.SubItems.Add(string.Format("{0:F1}", br.TBW));
        lvi.SubItems.Add(string.Format("{0} - {1}", bn.TBW_L, bn.TBW_H));

        // ECW %
        lvi = lstviewResultsList.Items.Add("Extra Cellular Water percentage");
        lvi.SubItems.Add((BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) ==
    BodystatAPI.BSDeviceFamily.BSQuadScan_Family ||
    BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) ==
    BodystatAPI.BSDeviceFamily.BSMDD_Family) ? string.Format("{0:F1}", br.ECWPerc) : "(MDD/Quad
    Only)");
        lvi.SubItems.Add((BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) ==
    BodystatAPI.BSDeviceFamily.BSQuadScan_Family ||
    BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) ==
    BodystatAPI.BSDeviceFamily.BSMDD_Family) ? string.Format("{0}", bn.ECWPerc_Norm) :
    "(MDD/Quad Only)");

        // ECW lt
        lvi = lstviewResultsList.Items.Add("Extra Cellular Water (in litres)");
        lvi.SubItems.Add((BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) ==
    BodystatAPI.BSDeviceFamily.BSQuadScan_Family ||
    BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) ==
    BodystatAPI.BSDeviceFamily.BSMDD_Family) ? string.Format("{0:F1}", br.ECW) : "(Quad Only)");
        lvi.SubItems.Add("N/A");

        // ICW %
        lvi = lstviewResultsList.Items.Add("Intra Cellular Water percentage");
        lvi.SubItems.Add((BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) ==
    BodystatAPI.BSDeviceFamily.BSQuadScan_Family ||
    BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) ==
    BodystatAPI.BSDeviceFamily.BSMDD_Family) ? string.Format("{0:F1}", br.ICWPerc) : "(MDD/Quad
    Only)");
        lvi.SubItems.Add((BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) ==
    BodystatAPI.BSDeviceFamily.BSQuadScan_Family ||
    BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) ==
    BodystatAPI.BSDeviceFamily.BSMDD_Family) ? string.Format("{0}", bn.ICWPerc_Norm) :
    "(MDD/Quad Only)");

        // ICW lt
        lvi = lstviewResultsList.Items.Add("Intra Cellular Water (in litres)");
        lvi.SubItems.Add((BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) ==
    BodystatAPI.BSDeviceFamily.BSQuadScan_Family ||
    BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) ==
    BodystatAPI.BSDeviceFamily.BSMDD_Family) ? string.Format("{0:F1}", br.ICW) : "(MDD/Quad
    Only)");
        lvi.SubItems.Add("N/A");

        // Body Cell Mass
        lvi = lstviewResultsList.Items.Add("Body Cell Mass");
        lvi.SubItems.Add(BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) ==
    BodystatAPI.BSDeviceFamily.BSQuadScan_Family ? string.Format("{0:F1}", br.BCM) : "(Quad
    Only)");
        lvi.SubItems.Add("N/A");

        // 3rd space water
        lvi = lstviewResultsList.Items.Add("3rd space water (in litres)");
        lvi.SubItems.Add(BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) ==
    BodystatAPI.BSDeviceFamily.BSQuadScan_Family ? string.Format("{0:F1}", br.ThirdSpace) :
    "(Quad Only)");
        lvi.SubItems.Add("N/A");

        // Nutrition index
        lvi = lstviewResultsList.Items.Add("Nutrition index");
        lvi.SubItems.Add(BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) ==
    BodystatAPI.BSDeviceFamily.BSQuadScan_Family ? string.Format("{0:F2}", br.Nutrition) :
    "(Quad Only)");
        lvi.SubItems.Add(BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) ==
    BodystatAPI.BSDeviceFamily.BSQuadScan_Family ? string.Format("{0:F2}", bn.Nutrition_Norm) :
    "(Quad Only)");

        // Prediction Marker (TM) - formerly known as Illness Marker (TM)
        lvi = lstviewResultsList.Items.Add("Prediction Marker (TM)");
        lvi.SubItems.Add(BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) ==
    BodystatAPI.BSDeviceFamily.BSQuadScan_Family ? string.Format("{0:F3}", br.Illness) : "(Quad
    Only)");
        lvi.SubItems.Add(BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) ==
    BodystatAPI.BSDeviceFamily.BSQuadScan_Family ? string.Format("{0:F3} - {1:F3}",
    bn.Illness_L, bn.Illness_H) : "(Quad Only)");

        // BMR kcal
        lvi = lstviewResultsList.Items.Add("Basal Metabolic Rate (in kcal)");
        lvi.SubItems.Add(string.Format("{0:F0}", br.BMR));
        lvi.SubItems.Add("N/A");

        // BMR kcal/kg
        lvi = lstviewResultsList.Items.Add("Basal Metabolic Rate per kilogram (in kcal/kg)");
        lvi.SubItems.Add(string.Format("{0:F1}", br.BMRkg));
        lvi.SubItems.Add("N/A");

        // EAR
        lvi = lstviewResultsList.Items.Add("Estimated Average Requirement (in kcal)");
        lvi.SubItems.Add(string.Format("{0:F0}", br.EstAvg));
        lvi.SubItems.Add("N/A");

        // BMI
        lvi = lstviewResultsList.Items.Add("Body Mass Index (BMI)");
        lvi.SubItems.Add(string.Format("{0:F1}", br.BMI));
        lvi.SubItems.Add(string.Format("{0} - {1}", bn.BMI_L, bn.BMI_H));

        // BFMI
        lvi = lstviewResultsList.Items.Add("Body Fat Mass Index (BFMI)");
        lvi.SubItems.Add((BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) ==
    BodystatAPI.BSDeviceFamily.BSQuadScan_Family ||
    BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) ==
    BodystatAPI.BSDeviceFamily.BSMDD_Family) ? string.Format("{0:F1}", br.BFMI) : "(MDD/Quad
    Only)");
        lvi.SubItems.Add((BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) ==
    BodystatAPI.BSDeviceFamily.BSQuadScan_Family ||
    BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) ==
    BodystatAPI.BSDeviceFamily.BSMDD_Family) ? string.Format("{0} - {1}", bn.BFMI_L, bn.BFMI_H)
    : "(MDD/Quad Only)");

        // FFMI
        lvi = lstviewResultsList.Items.Add("Fat Free Mass Index (FFMI)");
        lvi.SubItems.Add((BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) ==
    BodystatAPI.BSDeviceFamily.BSQuadScan_Family ||
    BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) ==
    BodystatAPI.BSDeviceFamily.BSMDD_Family) ? string.Format("{0:F1}", br.FFMI) : "(MDD/Quad
    Only)");
        lvi.SubItems.Add((BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) ==
    BodystatAPI.BSDeviceFamily.BSQuadScan_Family ||
    BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) ==
    BodystatAPI.BSDeviceFamily.BSMDD_Family) ? string.Format("{0} - {1}", bn.FFMI_L, bn.FFMI_H)
    : "(MDD/Quad Only)");

        // Waist/Hip
        lvi = lstviewResultsList.Items.Add("Waist/Hip ratio");
        lvi.SubItems.Add(string.Format("{0:F2}", br.WaistHip));
        lvi.SubItems.Add(string.Format("{0:F2}", bn.WaistHip_Norm));

        // Wellness marker
        lvi = lstviewResultsList.Items.Add("Wellness Marker (TM)");
        lvi.SubItems.Add(BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) ==
    BodystatAPI.BSDeviceFamily.BSMDD_Family ? string.Format("{0:F3}", br.Wellness) : "(MDD
    Only)");
```

```
    lvi.SubItems.Add(BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) ==
BodystatAPI.BSDeviceFamily.BSMDD_Family ? string.Format("{0:F3} - {0:F3}", bn.Wellness_L,
bn.Wellness_H) : "(MDD Only)");
}
```

4.2.1.1.2.2.4 ResultsForm.UseBSMeasurement Method

Allows the caller to specify a downloaded measurement structure to be use when initialising the dialog. Must be called before the form is loaded (displayed)

C++

```
void UseBSMeasurement(BodystatAPI.BodystatAPI.BSMeasurement bsm);
```

C#

```
public void UseBSMeasurement(BodystatAPI.BodystatAPI.BSMeasurement bsm);
```

Visual Basic

```
Public Function UseBSMeasurement(bsm As BodystatAPI.BodystatAPI.BSMeasurement) As void
```

Java

```
public void UseBSMeasurement(BodystatAPI.BodystatAPI.BSMeasurement bsm);
```

MATLAB

```
UseBSMeasurement
```

Body Source

```
public void UseBSMeasurement(BodystatAPI.BodystatAPI.BSMeasurement bsm)
{
    bsmMeasurement = bsm;
    bUseActualMeasurment = true;
}
```

4.2.2 Files

The following table lists files in this documentation.

Files

Name	Description
MainForm.cs (see page 216)	Bodystat VB.Net Sample (C) 2010-2012 Bodystat (Isle of Man) Ltd. All rights reserved. Use subject to license. Web: http://www.bodystat.com Email: info@bodystat.com Tel: +44 (0)1624 629 571 Fax: +44 (0)1624 611 544
ResultsForm.cs (see page 216)	Bodystat C# WinForms Sample (C) 2010-2014 Bodystat (Isle of Man) Ltd. All rights reserved. Use subject to license. Web: http://www.bodystat.com Email: info@bodystat.com Tel: +44 (0)1624 629 571 Fax: +44 (0)1624 611 544

Module

Bodystat C# WinForms Sample (see page 204)

4.2.2.1 MainForm.cs

Bodystat VB.Net Sample

(C) 2010-2012 Bodystat (Isle of Man) Ltd. All rights reserved.

Use subject to license.

Web: http://www.bodystat.com

Email: info@bodystat.com

Tel: +44 (0)1624 629 571

Fax: +44 (0)1624 611 544

Namespaces

Name	Description
BodystatCSWinForms (see page 205)	This is the main namespace for our sample project.

4.2.2.2 ResultsForm.cs

Bodystat C# WinForms Sample

(C) 2010-2014 Bodystat (Isle of Man) Ltd. All rights reserved.

Use subject to license.

Web: http://www.bodystat.com

Email: info@bodystat.com

Tel: +44 (0)1624 629 571

Fax: +44 (0)1624 611 544

Namespaces

Name	Description
BodystatCSWinForms (see page 205)	This is the main namespace for our sample project.

4.3 Bodystat Visual Basic .NET Sample

A sample application written in Visual Basic .NET based upon the Microsoft .NET Framework. Solutions are provided for Visual Basic .NET 2008 and 2010. You must have Visual Basic .NET 2008 or 2010 installed to work with this sample.

Namespaces

Name	Description
BodystatVBNetWinFormSample (see page 217)	This is the main namespace for our sample project.

4.3.1 BodystatVBNetWinFormSample Namespace

This is the main namespace for our sample project.

Module

Bodystat Visual Basic .NET Sample (see page 216)

Classes

	Name	Description
	FrmBodystatMain (see page 217)	Main sample dialog. Provides buttons utilising the various functions of the Bodystat API. A list control also shows data (test measurements) downloaded from the device.
	FrmBodystatResults (see page 237)	Results sample dialog. Displays the various parameters relating to a measurement (allowing direct input by the user), then utilises the Bodystat API to calculate detailed results and normals.

4.3.1.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	FrmBodystatMain (see page 217)	Main sample dialog. Provides buttons utilising the various functions of the Bodystat API. A list control also shows data (test measurements) downloaded from the device.
	FrmBodystatResults (see page 237)	Results sample dialog. Displays the various parameters relating to a measurement (allowing direct input by the user), then utilises the Bodystat API to calculate detailed results and normals.

4.3.1.1.1 FrmBodystatMain Class

Main sample dialog. Provides buttons utilising the various functions of the Bodystat API. A list control also shows data (test measurements) downloaded from the device.

Class Hierarchy

C++

```
class FrmBodystatMain;
```

C#

```
public class FrmBodystatMain;
```

Visual Basic

```
Public Class FrmBodystatMain
```

Java

```
public class FrmBodystatMain;
```

MATLAB

```
FrmBodystatMain
```

File

FrmBodystatMain.vb

FrmBodystatMain Fields

	Name	Description
	iTimeout (see page 219)	Bluetooth timeout multiplier. Initial value of 7 (equates to about 10 seconds). This is incremented upon each Bluetooth search. A maximum value of 45 is permitted (about 1 minute).

FrmBodystatMain Methods

	Name	Description
	btnBtAuthenticateButton_Click (see page 219)	Executes the Bluetooth Authenticate button action. Illustrates use of BSAuthenticateBTDevice from the Bodystat API
	btnBtAutoSetupButton_Click (see page 220)	Executes the Bluetooth Auto Setup button action. Illustrates use of BSAutoSetupBT from the Bodystat API
	btnBtDeviceInfoButton_Click (see page 220)	Executes the Bluetooth Device Info button action. Illustrates use of BSGetBTBodystatDevice from the Bodystat API
	btnBtSearchButton_Click (see page 221)	Executes the Bluetooth Search button action. Illustrates use of BSSearchBTDevices from the Bodystat API
	btnBtStackInfoButton_Click (see page 222)	Executes the Bluetooth Get Stack Info button action. Illustrates use of BSGetBTStackInfo from the Bodystat API
	btnBtStatusButton_Click (see page 222)	Executes the Bluetooth Get Status button action. Illustrates use of BSIsBTAvailable from the Bodystat API
	btnBtUnauthenticateButton_Click (see page 223)	Executes the Bluetooth Unauthenticate button action. Illustrates use of BSUnAuthenticateBTDevices from the Bodystat API
	btnComConnectButton_Click (see page 223)	Executes the communication Connect button action. Illustrates use of BSOpenComport from the Bodystat API
	btnComReadCalibrationTimeButton_Click (see page 224)	Executes the communication Read Calibration Time button action. Illustrates use of BSReadCalibrationTime from the Bodystat API
	btnComReadCurrentTimeButton_Click (see page 225)	Executes the communication Read Current Time button action. Illustrates use of BSReadCurrentTime from the Bodystat API
	btnComReadModelVersionButton_Click (see page 226)	Executes the communication Read Model Version button action. Illustrates use of BSReadModelVersion from the Bodystat API
	btnComReadPrinterAddrButton_Click (see page 227)	Executes the communication Read Printer Address button action. Illustrates use of BSReadPrinterAddress from the Bodystat API
	btnComReadProtocolInfoButton_Click (see page 228)	Executes the communication ReadProtocolInfo button action. Illustrates use of BSReadProtocolInfo from the Bodystat API
	btnComReadSerialNumberButton_Click (see page 229)	Executes the communication Read Serial Number button action. Illustrates use of BSReadSerialNumber from the Bodystat API
	btnComReadStoredTestDataButton_Click (see page 230)	Executes the communication Read Stored Test Data button action. Illustrates use of BSReadStoredTestData from the Bodystat API to download data
	btnComReadTestMeasurementButton_Click (see page 231)	Executes the communication Read Test Measurement button action. Performs an immediate test measurement on the device using the BSReadTestBodystat function from the Bodystat API
	btnComWriteCurrentTimeButton_Click (see page 232)	Executes the communication Write Current Time button action. Illustrates use of BSWriteCurrentTime from the Bodystat API
	btnResultsButton_Click (see page 233)	Executes the View Results button action. Opens the results form ready for more detailed analysis and normals
	FrmBodystatMain_Load (see page 234)	Prepare the controls on the sample dialog
	lstviewTestMeasurementList_DoubleClick (see page 235)	Executes the Measurements lists double click action. Opens the results form ready for more detailed analysis and normals
	PopulateMeasurementList (see page 235)	Populate the measurement list control with test measurement data from the API. Illustrates iteration through a BSRawData structure, working with data in a BSMeasurement structure and helper functions from the Bodystat API like BSGetDeviceModelName

4.3.1.1.1.1 FrmBodystatMain Fields

4.3.1.1.1.1.1 FrmBodystatMain.iTimeout Field

C++

```
UInt16 iTimeout;
```

C#

```
public UInt16 iTimeout;
```

Visual Basic

```
Public iTimeout As UInt16
```

Java

```
public UInt16 iTimeout;
```

MATLAB

```
iTimeout
```

Description

Bluetooth timeout multiplier. Initial value of 7 (equates to about 10 seconds). This is incremented upon each Bluetooth search. A maximum value of 45 is permitted (about 1 minute).

4.3.1.1.1.2 FrmBodystatMain Methods

4.3.1.1.1.2.1 FrmBodystatMain.btnBtAuthenticateButton_Click Method

Executes the Bluetooth Authenticate button action. Illustrates use of BSAuthenticateBTDevice from the Bodystat API

C++

```
btnBtAuthenticateButton_Click(System.Object sender, System.EventArgs e);
```

C#

```
private btnBtAuthenticateButton_Click(System.Object sender, System.EventArgs e);
```

Visual Basic

```
Private Sub btnBtAuthenticateButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnBtAuthenticateButton.Click
```

Java

```
private btnBtAuthenticateButton_Click(System.Object sender, System.EventArgs e);
```

MATLAB

```
btnBtAuthenticateButton_Click
```

Body Source

```
Private Sub btnBtAuthenticateButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnBtAuthenticateButton.Click
    Me.Cursor = Cursors.WaitCursor  ' Lengthy operation, show the user we are doing
something

    If (BodystatAPI.BodystatAPI.BSAuthenticateBTDevice(Me.Handle, iTimeout)) Then
        MsgBox("Device authenticated\nClick Device Info button for more information",
MsgBoxStyle.Information)
    Else
        MsgBox("Error no Bodystat device could be found or authenticated",
MsgBoxStyle.Information)
    End If

    Me.Cursor = Cursors.Default
End Sub
```

4.3.1.1.1.2.2 FrmBodystatMain.btnBtAutoSetupButton_Click Method

Executes the Bluetooth Auto Setup button action. Illustrates use of BSAutoSetupBT from the Bodystat API

C++

```
btnBtAutoSetupButton_Click(System.Object sender, System.EventArgs e);
```

C#

```
private btnBtAutoSetupButton_Click(System.Object sender, System.EventArgs e);
```

Visual Basic

```
Private Sub btnBtAutoSetupButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnBtAutoSetupButton.Click
```

Java

```
private btnBtAutoSetupButton_Click(System.Object sender, System.EventArgs e);
```

MATLAB

```
btnBtAutoSetupButton_Click
```

Body Source

```
Private Sub btnBtAutoSetupButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnBtAutoSetupButton.Click
    Dim strComPort As New StringBuilder(256)

    Me.Cursor = Cursors.WaitCursor  ' Lengthy operation, show the user we are doing
something

    ' Perform auto setup. User will be prompted as needed by the API
    ' Suitable for supported locales only. Non-supported locales will display messages in
English
    ' For non-supported locales you should use the search and authenticate functions
directly (silently, prompting the user yourself as required)
    If (BodystatAPI.BodystatAPI.BSAutoSetupBT(strComPort, strComPort.Capacity, True,
Me.Handle)) Then
        Dim strMsg As String
        strMsg = String.Format("Device setup complete." & vbCrLf & "Com Port: {0}" & vbCrLf
& "Click Device Info button for more information", strComPort.ToString())

        Me.txtComPortEdit.Text = strComPort.ToString()         ' Assign this port to the
edit box on the form

        MsgBox(strMsg, MsgBoxStyle.Information)
    Else
        MsgBox("Error no Bodystat device could be found or authenticated")
    End If

    Me.Cursor = Cursors.Default
End Sub
```

4.3.1.1.1.2.3 FrmBodystatMain.btnBtDeviceInfoButton_Click Method

Executes the Bluetooth Device Info button action. Illustrates use of BSGetBTBodystatDevice from the Bodystat API

C++

```
btnBtDeviceInfoButton_Click(System.Object sender, System.EventArgs e);
```

C#

```
private btnBtDeviceInfoButton_Click(System.Object sender, System.EventArgs e);
```

Visual Basic

```
Private Sub btnBtDeviceInfoButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnBtDeviceInfoButton.Click
```

Java

```
private btnBtDeviceInfoButton_Click(System.Object sender, System.EventArgs e);
```

MATLAB

```
btnBtDeviceInfoButton_Click
```

Body Source

```
Private Sub btnBtDeviceInfoButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnBtDeviceInfoButton.Click
    Dim strDeviceName As New StringBuilder(512)
    Dim strComPort As New StringBuilder(18)
    Dim strBDA As New StringBuilder(256)

    Me.Cursor = Cursors.WaitCursor  ' Lengthy operation (if not already searched), show the
user we are doing something

    ' Retrieve information about the Bodystat device
    Dim bSuccess As Boolean
    bSuccess = BodystatAPI.BodystatAPI.BSGetBTBodystatDevice(strDeviceName,
strDeviceName.Capacity, strBDA, strBDA.Capacity, strComPort, strComPort.Capacity, iTimeout)

    If (bSuccess) Then
        Dim strMsg As String
        strMsg = String.Format("Found Bodystat device." & vbCrLf & vbCrLf & "Device Name:
{0}" & vbCrLf & "Bluetooth Address: {1}" & vbCrLf & "Com Port: {2}", strDeviceName, strBDA,
IIf(strComPort.Length = 0, "<requires authentication>", strComPort))

        Me.txtComPortEdit.Text = strComPort.ToString()         ' Assign this port to the
edit box on the form

        MsgBox(strMsg, MsgBoxStyle.Information)
    Else
        MsgBox("Error no Bodystat device was found." & vbCrLf & "Click Search Devices
button to retry.")
    End If

    Me.Cursor = Cursors.Default
End Sub
```

4.3.1.1.1.2.4 FrmBodystatMain.btnBtSearchButton_Click Method

Executes the Bluetooth Search button action. Illustrates use of BSSearchBTDevices from the Bodystat API

C++

```
btnBtSearchButton_Click(System.Object sender, System.EventArgs e);
```

C#

```
private btnBtSearchButton_Click(System.Object sender, System.EventArgs e);
```

Visual Basic

```
Private Sub btnBtSearchButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnBtSearchButton.Click
```

Java

```
private btnBtSearchButton_Click(System.Object sender, System.EventArgs e);
```

MATLAB

```
btnBtSearchButton_Click
```

Body Source

```
Private Sub btnBtSearchButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnBtSearchButton.Click
    Dim iNewDevices As Integer = 0
    Dim iAuthenticatedDevices As Integer = 0

    Me.Cursor = Cursors.WaitCursor  ' Lengthy operation, show the user we are doing
something

    ' Begin the search...
    If (BodystatAPI.BodystatAPI.BSSearchBTDevices(iNewDevices, iAuthenticatedDevices, True,
Me.Handle, iTimeout)) Then
```

```
        MsgBox(String.Format("Search complete." & vbCrLf & "Found {0} new and {1}
previously authenticated Bodystat Bluetooth devices." & vbCrLf & "Search waited
approximately {2}s.", iNewDevices, iAuthenticatedDevices, iTimeout * 1.28),
MsgBoxStyle.Information)
    Else
        MsgBox("Error searching for Bluetooth devices")
    End If

    ' If no new devices were found, lets try extending the search time for next time around
    ' We can wait up to the maximum of value of 45 - about 1 minute
    ' Obviously lengthy waits like this need to be in a background thread in a real world
example
    If (iNewDevices = 0) Then
        iTimeout = Math.Min(iTimeout * 2, 45)
    End If

    Me.Cursor = Cursors.Default
End Sub
```

4.3.1.1.1.2.5 FrmBodystatMain.btnBtStackInfoButton_Click Method

Executes the Bluetooth Get Stack Info button action. Illustrates use of BSGetBTStackInfo from the Bodystat API

C++

```
btnBtStackInfoButton_Click(System.Object sender, System.EventArgs e);
```

C#

```
private btnBtStackInfoButton_Click(System.Object sender, System.EventArgs e);
```

Visual Basic

```
Private Sub btnBtStackInfoButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnBtStackInfoButton.Click
```

Java

```
private btnBtStackInfoButton_Click(System.Object sender, System.EventArgs e);
```

MATLAB

```
btnBtStackInfoButton_Click
```

Body Source

```
Private Sub btnBtStackInfoButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnBtStackInfoButton.Click
    Dim strInfo As New StringBuilder(1024)
    If (BodystatAPI.BodystatAPI.BSGetBTStackInfo(strInfo, strInfo.Capacity)) Then
        MsgBox(strInfo.ToString(), MsgBoxStyle.Information)
    Else
        MsgBox("Error: No compatible Bluetooth Stack is available in this pc." & vbCrLf &
"Either it is switched off, or it is not recognised by this software." & vbCrLf & "Only
Microsoft and Widcomm stacks are supported by Bodystat's SDK.")
    End If
End Sub
```

4.3.1.1.1.2.6 FrmBodystatMain.btnBtStatusButton_Click Method

Executes the Bluetooth Get Status button action. Illustrates use of BSIsBTAvailable from the Bodystat API

C++

```
btnBtStatusButton_Click(System.Object sender, System.EventArgs e);
```

C#

```
private btnBtStatusButton_Click(System.Object sender, System.EventArgs e);
```

Visual Basic

```
Private Sub btnBtStatusButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnBtStatusButton.Click
```

Java

```
private btnBtStatusButton_Click(System.Object sender, System.EventArgs e);
```

MATLAB

```
btnBtStatusButton_Click
```

Body Source

```
Private Sub btnBtStatusButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnBtStatusButton.Click
    If (BodystatAPI.BodystatAPI.BSIsBTAvailable()) Then
        MsgBox("Supported Bluetooth radio module is available!", MsgBoxStyle.Information)
    Else
        MsgBox("No compatible Bluetooth radio module is available in this pc." & vbCrLf &
"Either it is switched off, or it is not recognised by this software." & vbCrLf & "Only
Microsoft and Widcomm stacks are supported by Bodystat's SDK.")
    End If
End Sub
```

4.3.1.1.1.2.7 FrmBodystatMain.btnBtUnauthenticateButton_Click Method

Executes the Bluetooth Unauthenticate button action. Illustrates use of BSUnAuthenticateBTDevices from the Bodystat API

C++

```
btnBtUnauthenticateButton_Click(System.Object sender, System.EventArgs e);
```

C#

```
private btnBtUnauthenticateButton_Click(System.Object sender, System.EventArgs e);
```

Visual Basic

```
Private Sub btnBtUnauthenticateButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnBtUnauthenticateButton.Click
```

Java

```
private btnBtUnauthenticateButton_Click(System.Object sender, System.EventArgs e);
```

MATLAB

```
btnBtUnauthenticateButton_Click
```

Body Source

```
Private Sub btnBtUnauthenticateButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnBtUnauthenticateButton.Click
    Me.Cursor = Cursors.WaitCursor  ' Lengthy operation, show the user we are doing
something

    If (BodystatAPI.BodystatAPI.BSUnAuthenticateBTDevices(True, Me.Handle, iTimeout)) Then
        MsgBox("Devices authentication removed", MsgBoxStyle.Information)
    Else
        MsgBox("Error removing authentication")
    End If

    Me.Cursor = Cursors.Default
End Sub
```

4.3.1.1.1.2.8 FrmBodystatMain.btnComConnectButton_Click Method

Executes the communication Connect button action. Illustrates use of BSOpenComport from the Bodystat API

C++

```
btnComConnectButton_Click(System.Object sender, System.EventArgs e);
```

C#

```
private btnComConnectButton_Click(System.Object sender, System.EventArgs e);
```

Visual Basic

```
Private Sub btnComConnectButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnComConnectButton.Click
```

Java

```
private btnComConnectButton_Click(System.Object sender, System.EventArgs e);
```

MATLAB

```
btnComConnectButton_Click
```

Body Source

```
Private Sub btnComConnectButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnComConnectButton.Click
    Me.Cursor = Cursors.WaitCursor  ' Lengthy operation, show the user we are doing
something

    Dim strComPort As New StringBuilder(10)
    strComPort.Append(txtComPortEdit.Text)

    ' Open a com port to the device
    Dim bsErr As BSError
    bsErr = BodystatAPI.BodystatAPI.BSOpenComport(strComPort, strComPort.Capacity)
    If (bsErr = BSError.NoError) Then

        bsErr = BodystatAPI.BodystatAPI.BSConnect()
        If (bsErr = BSError.NoError) Then
            MsgBox("Connected to device successfully.", MsgBoxStyle.Information)
        Else
            MsgBox("Error connecting to device.")
        End If

        ' Close the com port
        BodystatAPI.BodystatAPI.BSCloseComport()
    Else
        MsgBox("Error opening com port.")
    End If

    Me.Cursor = Cursors.Default
End Sub
```

4.3.1.1.1.2.9 FrmBodystatMain.btnComReadCalibrationTimeButton_Click Method

Executes the communication Read Calibration Time button action. Illustrates use of BSReadCalibrationTime from the Bodystat API

C++

```
btnComReadCalibrationTimeButton_Click(System.Object sender, System.EventArgs e);
```

C#

```
private btnComReadCalibrationTimeButton_Click(System.Object sender, System.EventArgs e);
```

Visual Basic

```
Private Sub btnComReadCalibrationTimeButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnComReadCalibrationTimeButton.Click
```

Java

```
private btnComReadCalibrationTimeButton_Click(System.Object sender, System.EventArgs e);
```

MATLAB

```
btnComReadCalibrationTimeButton_Click
```

Body Source

```
Private Sub btnComReadCalibrationTimeButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnComReadCalibrationTimeButton.Click
    Me.Cursor = Cursors.WaitCursor  ' Lengthy operation, show the user we are doing
something

    Dim strComPort As New StringBuilder(10)
    strComPort.Append(txtComPortEdit.Text)

    ' Open a com port to the device
    Dim bsErr As BSError
    bsErr = BodystatAPI.BodystatAPI.BSOpenComport(strComPort, strComPort.Capacity)
    If (bsErr = BSError.NoError) Then
```

```
        ' Connect to the device
        bsErr = BodystatAPI.BodystatAPI.BSConnect()
        If (bsErr = BSError.NoError) Then

            Dim dateTime As Date

            ' Read the calibration date from the device
            bsErr = BodystatAPI.BodystatAPI.BSReadCalibrationTime(dateTime)
            If (bsErr = BSError.NoError) Then

                Dim strMsg As String
                strMsg = String.Format("Queried device successfully" & vbCrLf &
"Calibration Date/Time: {0}", dateTime.ToString())
                MsgBox(strMsg, MsgBoxStyle.Information)

            Else
                MsgBox("Error querying calibration date/time from device." & vbCrLf & "Note
that legacy devices do store their calibration date electronically.")
            End If

        Else
            MsgBox("Error connecting to device.")
        End If

        ' Close the com port
        BodystatAPI.BodystatAPI.BSCloseComport()
    Else
        MsgBox("Error opening com port.")
    End If

    Me.Cursor = Cursors.Default
End Sub
```

4.3.1.1.1.2.10 FrmBodystatMain.btnComReadCurrentTimeButton_Click Method

Executes the communication Read Current Time button action. Illustrates use of BSReadCurrentTime from the Bodystat API

C++

```
btnComReadCurrentTimeButton_Click(System.Object sender, System.EventArgs e);
```

C#

```
private btnComReadCurrentTimeButton_Click(System.Object sender, System.EventArgs e);
```

Visual Basic

```
Private Sub btnComReadCurrentTimeButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnComReadCurrentTimeButton.Click
```

Java

```
private btnComReadCurrentTimeButton_Click(System.Object sender, System.EventArgs e);
```

MATLAB

```
btnComReadCurrentTimeButton_Click
```

Body Source

```
Private Sub btnComReadCurrentTimeButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnComReadCurrentTimeButton.Click
    Me.Cursor = Cursors.WaitCursor  ' Lengthy operation, show the user we are doing
something

    Dim strComPort As New StringBuilder(10)
    strComPort.Append(txtComPortEdit.Text)

    ' Open a com port to the device
    Dim bsErr As BSError
    bsErr = BodystatAPI.BodystatAPI.BSOpenComport(strComPort, strComPort.Capacity)
    If (bsErr = BSError.NoError) Then

        ' Connect to the device
        bsErr = BodystatAPI.BodystatAPI.BSConnect()
        If (bsErr = BSError.NoError) Then

            Dim dateTime As Date

            ' Read the current time from the internal clock of the device
            bsErr = BodystatAPI.BodystatAPI.BSReadCurrentTime(dateTime, -1)
            If (bsErr = BSError.NoError) Then

                Dim strMsg As String
                strMsg = String.Format("Queried device successfully" & vbCrLf & "Current
Date/Time: {0}", dateTime.ToString())
                MsgBox(strMsg, MsgBoxStyle.Information)

            Else
                MsgBox("Error querying current date/time from device." & vbCrLf & "Not all
models have an inbuilt real time clock.")
            End If

        Else
            MsgBox("Error connecting to device.")
        End If

        ' Close the com port
        BodystatAPI.BodystatAPI.BSCloseComport()
    Else
        MsgBox("Error opening com port.")
    End If

    Me.Cursor = Cursors.Default
End Sub
```

4.3.1.1.1.2.11 FrmBodystatMain.btnComReadModelVersionButton_Click Method

Executes the communication Read Model Version button action. Illustrates use of BSReadModelVersion from the Bodystat API

C++

```
btnComReadModelVersionButton_Click(System.Object sender, System.EventArgs e);
```

C#

```
private btnComReadModelVersionButton_Click(System.Object sender, System.EventArgs e);
```

Visual Basic

```
Private Sub btnComReadModelVersionButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnComReadModelVersionButton.Click
```

Java

```
private btnComReadModelVersionButton_Click(System.Object sender, System.EventArgs e);
```

MATLAB

```
btnComReadModelVersionButton_Click
```

Body Source

```
Private Sub btnComReadModelVersionButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnComReadModelVersionButton.Click
    Me.Cursor = Cursors.WaitCursor  ' Lengthy operation, show the user we are doing
something

    Dim strComPort As New StringBuilder(10)
    strComPort.Append(txtComPortEdit.Text)

    ' Open a com port to the device
    Dim bsErr As BSError
    bsErr = BodystatAPI.BodystatAPI.BSOpenComport(strComPort, strComPort.Capacity)
    If (bsErr = BSError.NoError) Then

        ' Connect to the device
        bsErr = BodystatAPI.BodystatAPI.BSConnect()
        If (bsErr = BSError.NoError) Then

            Dim iModel As BSDeviceModel
            Dim byMajor As Byte
            Dim byMinor As Byte
            Dim byPsoc2Version As Byte
            Dim byEepromVersion As Byte
            Dim byBluetoothInfo As Byte

            ' Read Model & Version information from the device
            bsErr = BodystatAPI.BodystatAPI.BSReadModelVersion(iModel, byMajor, byMinor,
byPsoc2Version, byEepromVersion, byBluetoothInfo)
            If (bsErr = BSError.NoError) Then

                Dim strMsg As String
                strMsg = String.Format("Queried device successfully" & vbCrLf & vbCrLf &
"Device Type: {0}" & vbCrLf & "Firmware Version: {1}.{2}.{3}.{4}", iModel, byMajor,
byMinor, byPsoc2Version, byEepromVersion)
                MsgBox(strMsg, MsgBoxStyle.Information)

            Else
                MsgBox("Error querying model information from device.")
            End If

        Else
            MsgBox("Error connecting to device.")
        End If

        ' Close the com port
        BodystatAPI.BodystatAPI.BSCloseComport()
    Else
        MsgBox("Error opening com port.")
    End If

    Me.Cursor = Cursors.Default
End Sub
```

4.3.1.1.1.2.12 FrmBodystatMain.btnComReadPrinterAddrButton_Click Method

Executes the communication Read Printer Address button action. Illustrates use of BSReadPrinterAddress from the Bodystat API

C++

```
btnComReadPrinterAddrButton_Click(System.Object sender, System.EventArgs e);
```

C#

```
private btnComReadPrinterAddrButton_Click(System.Object sender, System.EventArgs e);
```

Visual Basic

```
Private Sub btnComReadPrinterAddrButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnComReadPrinterAddrButton.Click
```

Java

```
private btnComReadPrinterAddrButton_Click(System.Object sender, System.EventArgs e);
```

MATLAB

```
btnComReadPrinterAddrButton_Click
```

Body Source

```
Private Sub btnComReadPrinterAddrButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnComReadPrinterAddrButton.Click
    Me.Cursor = Cursors.WaitCursor  ' Lengthy operation, show the user we are doing
something

    Dim strComPort As New StringBuilder(10)
    strComPort.Append(txtComPortEdit.Text)

    ' Open a com port to the device
    Dim bsErr As BSError
    bsErr = BodystatAPI.BodystatAPI.BSOpenComport(strComPort, strComPort.Capacity)
    If (bsErr = BSError.NoError) Then

        ' Connect to the device
        bsErr = BodystatAPI.BodystatAPI.BSConnect()
        If (bsErr = BSError.NoError) Then

            Dim strPrinterAddr As New StringBuilder(16)

            ' Read the address of the printer the device is paired to for direct printing
            bsErr = BodystatAPI.BodystatAPI.BSReadPrinterAddress(strPrinterAddr,
strPrinterAddr.Capacity)
            If (bsErr = BSError.NoError) Then

                Dim strMsg As String
                strMsg = String.Format("Queried device successfully" & vbCrLf & "Printer
Address: {0}", strPrinterAddr.ToString())
                MsgBox(strMsg, MsgBoxStyle.Information)

            Else
                MsgBox("Error querying printer address from device." & vbCrLf & "Not all
models support direct printing.")
            End If

        Else
            MsgBox("Error connecting to device.")
        End If

        ' Close the com port
        BodystatAPI.BodystatAPI.BSCloseComport()
    Else
        MsgBox("Error opening com port.")
    End If

    Me.Cursor = Cursors.Default
End Sub
```

4.3.1.1.1.2.13 FrmBodystatMain.btnComReadProtocolInfoButton_Click Method

Executes the communication ReadProtocolInfo button action. Illustrates use of BSReadProtocolInfo from the Bodystat API

C++

```
btnComReadProtocolInfoButton_Click(System.Object sender, System.EventArgs e);
```

C#

```
private btnComReadProtocolInfoButton_Click(System.Object sender, System.EventArgs e);
```

Visual Basic

```
Private Sub btnComReadProtocolInfoButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnComReadProtocolInfoButton.Click
```

Java

```
private btnComReadProtocolInfoButton_Click(System.Object sender, System.EventArgs e);
```

MATLAB

```
btnComReadProtocolInfoButton_Click
```

Body Source

```
Private Sub btnComReadProtocolInfoButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnComReadProtocolInfoButton.Click
    Me.Cursor = Cursors.WaitCursor  ' Lengthy operation, show the user we are doing
something

    Dim strComPort As New StringBuilder(10)
    strComPort.Append(txtComPortEdit.Text)

    ' Open a com port to the device
    Dim bsErr As BSError
    bsErr = BodystatAPI.BodystatAPI.BSOpenComport(strComPort, strComPort.Capacity)
```

```
        If (bsErr = BSError.NoError) Then

            ' Connect to the device
            bsErr = BodystatAPI.BodystatAPI.BSConnect()
            If (bsErr = BSError.NoError) Then

                Dim byProtocolVersion As Byte
                Dim byDataVersion As Byte
                Dim byAuxInfo As Byte

                ' Read Protocol Information from the device
                bsErr = BodystatAPI.BodystatAPI.BSReadProtocolInfo(byProtocolVersion,
byDataVersion, byAuxInfo)
                If (bsErr = BSError.NoError) Then

                    Dim strMsg As String
                    strMsg = String.Format("Queried device successfully" & vbCrLf & "Version:
{0}" & vbCrLf & "Data Version: {1}" & vbCrLf & "Aux Info: {2}", byProtocolVersion,
byDataVersion, byAuxInfo)
                    MsgBox(strMsg, MsgBoxStyle.Information)

                Else
                    MsgBox("Error querying protocol information from device.")
                End If

            Else
                MsgBox("Error connecting to device.")
            End If

            ' Close the com port
            BodystatAPI.BodystatAPI.BSCloseComport()
        Else
            MsgBox("Error opening com port.")
        End If

        Me.Cursor = Cursors.Default
    End Sub
```

4.3.1.1.1.2.14 FrmBodystatMain.btnComReadSerialNumberButton_Click Method

Executes the communication Read Serial Number button action. Illustrates use of BSReadSerialNumber from the Bodystat API

C++

```
btnComReadSerialNumberButton_Click(System.Object sender, System.EventArgs e);
```

C#

```
private btnComReadSerialNumberButton_Click(System.Object sender, System.EventArgs e);
```

Visual Basic

```
Private Sub btnComReadSerialNumberButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnComReadSerialNumberButton.Click
```

Java

```
private btnComReadSerialNumberButton_Click(System.Object sender, System.EventArgs e);
```

MATLAB

```
btnComReadSerialNumberButton_Click
```

Body Source

```
Private Sub btnComReadSerialNumberButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnComReadSerialNumberButton.Click
    Me.Cursor = Cursors.WaitCursor  ' Lengthy operation, show the user we are doing
something

    Dim strComPort As New StringBuilder(10)
    strComPort.Append(txtComPortEdit.Text)

    ' Open a com port to the device
    Dim bsErr As BSError
    bsErr = BodystatAPI.BodystatAPI.BSOpenComport(strComPort, strComPort.Capacity)
    If (bsErr = BSError.NoError) Then

        ' Connect to the device
        bsErr = BodystatAPI.BodystatAPI.BSConnect()
        If (bsErr = BSError.NoError) Then

            Dim ulSerialNumber As UInteger

            ' Read the unique serial number from the device
            bsErr = BodystatAPI.BodystatAPI.BSReadSerialNumber(ulSerialNumber)
            If (bsErr = BSError.NoError) Then

                Dim strMsg As String
                strMsg = String.Format("Queried device successfully" & vbCrLf & "Serial
Number: {0}", ulSerialNumber)
                MsgBox(strMsg, MsgBoxStyle.Information)

            Else
                MsgBox("Error querying serial number from device.")
            End If

        Else
            MsgBox("Error connecting to device.")
        End If

        ' Close the com port
        BodystatAPI.BodystatAPI.BSCloseComport()
    Else
        MsgBox("Error opening com port.")
    End If

    Me.Cursor = Cursors.Default
End Sub
```

4.3.1.1.1.2.15 FrmBodystatMain.btnComReadStoredTestDataButton_Click Method

Executes the communication Read Stored Test Data button action. Illustrates use of BSReadStoredTestData from the Bodystat API to download data

C++

```
btnComReadStoredTestDataButton_Click(System.Object sender, System.EventArgs e);
```

C#

```
private btnComReadStoredTestDataButton_Click(System.Object sender, System.EventArgs e);
```

Visual Basic

```
Private Sub btnComReadStoredTestDataButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnComReadStoredTestDataButton.Click
```

Java

```
private btnComReadStoredTestDataButton_Click(System.Object sender, System.EventArgs e);
```

MATLAB

```
btnComReadStoredTestDataButton_Click
```

Body Source

```
Private Sub btnComReadStoredTestDataButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnComReadStoredTestDataButton.Click
    Me.Cursor = Cursors.WaitCursor  ' Lengthy operation, show the user we are doing
something

    Dim strComPort As New StringBuilder(10)
    strComPort.Append(txtComPortEdit.Text)

    ' Open a com port to the device
    Dim bsErr As BSError
    bsErr = BodystatAPI.BodystatAPI.BSOpenComport(strComPort, strComPort.Capacity)
    If (bsErr = BSError.NoError) Then

        ' Connect to the device
        bsErr = BodystatAPI.BodystatAPI.BSConnect()
        If (bsErr = BSError.NoError) Then

            ' Need to allocate our raw data structure with precisely 100 records
            Dim m_bsRawData As New BodystatAPI.BodystatAPI.BSRawData(99)

            ' Download test measurement data from the device
            bsErr = BodystatAPI.BodystatAPI.BSReadStoredTestData(m_bsRawData)
            If (bsErr = BSError.NoError) Then

                Dim strMsg As String
                strMsg = String.Format("Download complete" & vbCrLf & "Retrieved {0}
records" & vbCrLf & "First Test Number: {1}", m_bsRawData.iTotalNumRecs,
m_bsRawData.ulFirstTestNum)

                ' Parse the data and place it in the measurement list control
                PopulateMeasurementList(m_bsRawData)

                MsgBox(strMsg, MsgBoxStyle.Information)

            Else
                MsgBox("Error downloading stored test data from the device.")
            End If

        Else
            MsgBox("Error connecting to device.")
        End If

        ' Close the com port
        BodystatAPI.BodystatAPI.BSCloseComport()
    Else
        MsgBox("Error opening com port.")
    End If

    Me.Cursor = Cursors.Default
End Sub
```

4.3.1.1.1.2.16 FrmBodystatMain.btnComReadTestMeasurementButton_Click Method

Executes the communication Read Test Measurement button action. Performs an immediate test measurement on the device using the BSReadTestBodystat function from the Bodystat API

C++

```
btnComReadTestMeasurementButton_Click(System.Object sender, System.EventArgs e);
```

C#

```
private btnComReadTestMeasurementButton_Click(System.Object sender, System.EventArgs e);
```

Visual Basic

```
Private Sub btnComReadTestMeasurementButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnComReadTestMeasurementButton.Click
```

Java

```
private btnComReadTestMeasurementButton_Click(System.Object sender, System.EventArgs e);
```

MATLAB

```
btnComReadTestMeasurementButton_Click
```

Body Source

```
Private Sub btnComReadTestMeasurementButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnComReadTestMeasurementButton.Click
    Me.Cursor = Cursors.WaitCursor  ' Lengthy operation, show the user we are doing
something

    Dim strComPort As New StringBuilder(10)
    strComPort.Append(txtComPortEdit.Text)
```

```
      ' Open a com port to the device
      Dim bsErr As BSError
      bsErr = BodystatAPI.BodystatAPI.BSOpenComport(strComPort, strComPort.Capacity)
      If (bsErr = BSError.NoError) Then

          ' Connect to the device
          bsErr = BodystatAPI.BodystatAPI.BSConnect()
          If (bsErr = BSError.NoError) Then

              Dim iZ5, iZ50, iZ100, iZ200, iR50 As Integer
              Dim fX50, fPA50 As Single

              ' Perform the test measurement on the device and get the raw electrical
  bio-impedance results
              bsErr = BodystatAPI.BodystatAPI.BSReadTestBodystat(iZ5, iZ50, iZ100, iZ200,
  iR50, fX50, fPA50)
              If (bsErr = BSError.NoError) Then

                  Dim strMsg As String
                  strMsg = String.Format("Measurement complete" & vbCrLf & _
                                         "Impedance 5kHz: {0}" & vbCrLf & _
                                         "Impedance 50kHz: {1}" & vbCrLf & _
                                         "Impedance 100kHz: {2}" & vbCrLf & _
                                         "Impedance 200kHz: {3}" & vbCrLf & _
                                         "Resistance 50kHz: {4}" & vbCrLf & _
                                         "Reactance 50kHz: {5:F1}" & vbCrLf & _
                                         "Phase Angle 50kHz: {6:F1}", _
                                         iZ5, iZ50, iZ100, iZ200, iR50, fX50, fPA50)

                  MsgBox(strMsg, MsgBoxStyle.Information)

              Else
                  MsgBox("Error performing test measurement on the device.")
              End If

          Else
              MsgBox("Error connecting to device.")
          End If

          ' Close the com port
          BodystatAPI.BodystatAPI.BSCloseComport()
      Else
          MsgBox("Error opening com port.")
      End If

      Me.Cursor = Cursors.Default
  End Sub
```

4.3.1.1.1.2.17 FrmBodystatMain.btnComWriteCurrentTimeButton_Click Method

Executes the communication Write Current Time button action. Illustrates use of BSWriteCurrentTime from the Bodystat API

C++

```
btnComWriteCurrentTimeButton_Click(System.Object sender, System.EventArgs e);
```

C#

```
private btnComWriteCurrentTimeButton_Click(System.Object sender, System.EventArgs e);
```

Visual Basic

```
Private Sub btnComWriteCurrentTimeButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnComWriteCurrentTimeButton.Click
```

Java

```
private btnComWriteCurrentTimeButton_Click(System.Object sender, System.EventArgs e);
```

MATLAB

```
btnComWriteCurrentTimeButton_Click
```

Body Source

```
Private Sub btnComWriteCurrentTimeButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnComWriteCurrentTimeButton.Click
    Me.Cursor = Cursors.WaitCursor  ' Lengthy operation, show the user we are doing
something

    Dim strComPort As New StringBuilder(10)
    strComPort.Append(txtComPortEdit.Text)

    ' Open a com port to the device
    Dim bsErr As BSError
    bsErr = BodystatAPI.BodystatAPI.BSOpenComport(strComPort, strComPort.Capacity)
    If (bsErr = BSError.NoError) Then

        ' Connect to the device
        bsErr = BodystatAPI.BodystatAPI.BSConnect()
        If (bsErr = BSError.NoError) Then

            ' Write a specific time to the device
            'Dim dtNow As DateTime = DateTime.Now().ToUniversalTime()
            'bsErr = BodystatAPI.BodystatAPI.BSWriteCurrentTimeAs(dtNow)

            ' Write the current local PC time to the internal clock of the device
            bsErr = BodystatAPI.BodystatAPI.BSWriteCurrentTime()
            If (bsErr = BSError.NoError) Then
                MsgBox("Device updated successfully" & vbCrLf & "Current date/time
updated.", MsgBoxStyle.Information)
            Else
                MsgBox("Error writing current date/time to device." & vbCrLf & "Not all
models have an inbuilt real time clock.")
            End If

        Else
            MsgBox("Error connecting to device.")
        End If

        ' Close the com port
        BodystatAPI.BodystatAPI.BSCloseComport()
    Else
        MsgBox("Error opening com port.")
    End If

    Me.Cursor = Cursors.Default
End Sub
```

4.3.1.1.1.2.18 FrmBodystatMain.btnResultsButton_Click Method

Executes the View Results button action. Opens the results form ready for more detailed analysis and normals

C++

```
btnResultsButton_Click(System.Object sender, System.EventArgs e);
```

C#

```
private btnResultsButton_Click(System.Object sender, System.EventArgs e);
```

Visual Basic

```
Private Sub btnResultsButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnResultsButton.Click
```

Java

```
private btnResultsButton_Click(System.Object sender, System.EventArgs e);
```

MATLAB

```
btnResultsButton_Click
```

Body Source

```
Private Sub btnResultsButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnResultsButton.Click
```

```
    Dim frmResults As New FrmBodystatResults()    ' Initialise the results dialog

    ' Populate the results dialog with the details of the selected measurement
    ' If no item is selected, we simply show the results dialog with its default values
    If (lstviewTestMeasurementList.SelectedItems.Count() = 1) Then
        Dim pBSM As BodystatAPI.BodystatAPI.BSMeasurement =
lstviewTestMeasurementList.SelectedItems.Item(0).Tag
        frmResults.UseBSMeasurement(pBSM)
    Else
        MsgBox("No measurement selected, an example measurement will be shown instead.",
MsgBoxStyle.Information)
    End If

    ' Show the results dialog
    frmResults.ShowDialog()

End Sub
```

4.3.1.1.1.2.19 FrmBodystatMain.FrmBodystatMain_Load Method

Prepare the controls on the sample dialog

C++

```
FrmBodystatMain_Load(System.Object sender, System.EventArgs e);
```

C#

```
private FrmBodystatMain_Load(System.Object sender, System.EventArgs e);
```

Visual Basic

```
Private Sub FrmBodystatMain_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
```

Java

```
private FrmBodystatMain_Load(System.Object sender, System.EventArgs e);
```

MATLAB

```
FrmBodystatMain_Load
```

Body Source

```
Private Sub FrmBodystatMain_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

    ' Add columns to the test measurement list ready for download of data
    Me.lstviewTestMeasurementList.Columns.Add("Test No", 50)
    Me.lstviewTestMeasurementList.Columns.Add("Test Date/Time", 130)
    Me.lstviewTestMeasurementList.Columns.Add("Model", 140)
    Me.lstviewTestMeasurementList.Columns.Add("Serial No", 75)
    Me.lstviewTestMeasurementList.Columns.Add("Gender", 75)
    Me.lstviewTestMeasurementList.Columns.Add("Age", 75)
    Me.lstviewTestMeasurementList.Columns.Add("Height", 75)
    Me.lstviewTestMeasurementList.Columns.Add("Weight", 75)
    Me.lstviewTestMeasurementList.Columns.Add("Activity", 75)
    Me.lstviewTestMeasurementList.Columns.Add("Waist", 75)
    Me.lstviewTestMeasurementList.Columns.Add("Hip", 75)
    Me.lstviewTestMeasurementList.Columns.Add("Imp 5 kHz", 75)
    Me.lstviewTestMeasurementList.Columns.Add("Imp 50 kHz", 75)
    Me.lstviewTestMeasurementList.Columns.Add("Imp 100 kHz", 75)
    Me.lstviewTestMeasurementList.Columns.Add("Imp 200 kHz", 75)
    Me.lstviewTestMeasurementList.Columns.Add("Res 50 kHz", 75)
    Me.lstviewTestMeasurementList.Columns.Add("Reac 50 kHz", 75)
    Me.lstviewTestMeasurementList.Columns.Add("Phase 50 kHz", 80)

    iTimeout = 7       ' Initial Bluetooth timeout multiplier of 7 (equates to about 10
seconds). This is incremented upon each Bluetooth search. A maximum value of 45 is
permitted (about 1 minute).
End Sub
```

4.3.1.1.1.2.20 FrmBodystatMain.lstviewTestMeasurementList_DoubleClick Method

Executes the Measurements lists double click action. Opens the results form ready for more detailed analysis and normals

C++

```
lstviewTestMeasurementList_DoubleClick(System.Object sender, System.EventArgs e);
```

C#

```
private lstviewTestMeasurementList_DoubleClick(System.Object sender, System.EventArgs e);
```

Visual Basic

```
Private Sub lstviewTestMeasurementList_DoubleClick(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles lstviewTestMeasurementList.DoubleClick
```

Java

```
private lstviewTestMeasurementList_DoubleClick(System.Object sender, System.EventArgs e);
```

MATLAB

```
lstviewTestMeasurementList_DoubleClick
```

Body Source

```
Private Sub lstviewTestMeasurementList_DoubleClick(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles lstviewTestMeasurementList.DoubleClick
    btnResultsButton_Click(sender, e)       ' Display the selected result
End Sub
```

4.3.1.1.1.2.21 FrmBodystatMain.PopulateMeasurementList Method

Populate the measurement list control with test measurement data from the API. Illustrates iteration through a BSRawData structure, working with data in a BSMeasurement structure and helper functions from the Bodystat API like BSGetDeviceModelName

C++

```
PopulateMeasurementList(BodystatAPI.BodystatAPI.BSRawData ^ pRawData);
```

C#

```
public PopulateMeasurementList(ref BodystatAPI.BodystatAPI.BSRawData pRawData);
```

Visual Basic

```
Public Sub PopulateMeasurementList(ByRef pRawData As BodystatAPI.BodystatAPI.BSRawData)
```

Java

```
public PopulateMeasurementList(BodystatAPI.BodystatAPI.BSRawData pRawData);
```

MATLAB

```
PopulateMeasurementList
```

Parameters

Parameters	Description
ByRef pRawData As BodystatAPI.BodystatAPI.BSRawData	[in] RawData to be parsed.

Body Source

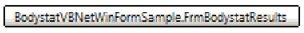
```
Public Sub PopulateMeasurementList(ByRef pRawData As BodystatAPI.BodystatAPI.BSRawData)

    ' Clear any previous items in the list
    Me.lstviewTestMeasurementList.Items.Clear()

    Dim strItem As String
    Dim strBuilder As StringBuilder

    Dim lvi As System.Windows.Forms.ListViewItem
    Dim i As Integer

    ' Process each record in the structure
```

```
        For i = 0 To pRawData.iTotalNumRecs - 1

            ' Test no
            strItem = String.Format("{0}", i + pRawData.ulFirstTestNum)
            lvi = lstviewTestMeasurementList.Items.Insert(i, strItem)
            lvi.Tag = pRawData.record(i)    ' Assign this BSMeasurement to the listview item
(so we can reference it later)

            ' Test date/time
            lvi.SubItems.Add(pRawData.record(i).tTestDate.ToString())

            ' Device model
            strBuilder = New StringBuilder(32)
            BodystatAPI.BodystatAPI.BSGetDeviceModelName(pRawData.record(i).iDeviceModel,
strBuilder, strBuilder.Capacity)
            lvi.SubItems.Add(strBuilder.ToString())

            ' Device serial number
            strItem = String.Format("{0}", pRawData.record(i).ulDeviceSerialNumber)
            lvi.SubItems.Add(strItem)

            ' Gender
            lvi.SubItems.Add(IIf(pRawData.record(i).iGender = BSGender.BSMale, "Male",
"Female"))

            ' Age
            strItem = String.Format("{0}", pRawData.record(i).iAge)
            lvi.SubItems.Add(strItem)

            ' Height
            strItem = String.Format("{0} cm", pRawData.record(i).iHeight)
            lvi.SubItems.Add(strItem)

            ' Weight
            strItem = String.Format("{0:F1} kg", pRawData.record(i).fWeight)
            lvi.SubItems.Add(strItem)

            ' Activity
            Select Case pRawData.record(i).iActivity
                Case 1
                    strItem = "Very Low"
                Case 2
                    strItem = "Low/Medium"
                Case 3
                    strItem = "Medium"
                Case 4
                    strItem = "Medium/High"
                Case 5
                    strItem = "Very High"
                Case Else
                    strItem = "Unknown"
            End Select
            lvi.SubItems.Add(strItem)

            ' Waist
            strItem = String.Format("{0} cm", pRawData.record(i).iWaist)
            lvi.SubItems.Add(strItem)

            ' Hip
            strItem = String.Format("{0} cm", pRawData.record(i).iHip)
            lvi.SubItems.Add(strItem)

            ' Z5 (Impedance at 5 kHz)
            strItem = String.Format("{0} ohm", pRawData.record(i).iZ_5kHz)
            lvi.SubItems.Add(strItem)

            ' Z50 (Impedance at 50 kHz)
            strItem = String.Format("{0} ohm", pRawData.record(i).iZ_50kHz)
            lvi.SubItems.Add(strItem)

            ' Z100 (Impedance at 100 kHz)
            strItem = String.Format("{0} ohm", pRawData.record(i).iZ_100kHz)
            lvi.SubItems.Add(strItem)

            ' Z200 (Impedance at 200 kHz)
            strItem = String.Format("{0} ohm", pRawData.record(i).iZ_200kHz)
            lvi.SubItems.Add(strItem)

            ' R50 (Resistance at 50 kHz)
            strItem = String.Format("{0} ohm", pRawData.record(i).iR_50kHz)
            lvi.SubItems.Add(strItem)

            ' X50 (Reactance at 50 kHz)
            strItem = String.Format("{0:F1} ohm", pRawData.record(i).fX_50kHz)
            lvi.SubItems.Add(strItem)

            ' PA50 (Phase Angle at 50 kHz)
            strItem = String.Format("{0:F1} deg", pRawData.record(i).fPA_50kHz)
            lvi.SubItems.Add(strItem)

        Next i
    End Sub
```

4.3.1.1.2 FrmBodystatResults Class

Results sample dialog. Displays the various parameters relating to a measurement (allowing direct input by the user), then utilises the Bodystat API to calculate detailed results and normals.

Class Hierarchy



C++

```
class FrmBodystatResults;
```

C#

```
public class FrmBodystatResults;
```

Visual Basic

```
Public Class FrmBodystatResults
```

Java

```
public class FrmBodystatResults;
```

MATLAB

```
FrmBodystatResults
```

File

FrmBodystatResults.vb

FrmBodystatResults Fields

	Name	Description
◆	bsmMeasurement (↗ see page 238)	Base the results dialog on this downloaded measurement
◆	bUseActualMeasurment (↗ see page 238)	Determines whether the dialog should use the measurement values or sample values
◆	dtTestDate (↗ see page 238)	Test date/time, not presently shown on the form (but may be passed to clipboard data)

FrmBodystatResults Methods

	Name	Description
◆	btnCalculateResultsButton_Click (↗ see page 239)	Executes the Calculate Results button action. Uses the Bodystat API to calculate the results and normals for this test measurement.
◆	btnCopytoclipboardButton_Click (↗ see page 240)	Implements the Copy To Clipboard button action. Copies the various parameters relating to a measurement directly to the clipboard

	FrmBodystatResults_Load (see page 242)	Initialise the dialog controls with their content
	InitialiseFromMeasurement (see page 242)	Initialise the dialog controls with a values from an actual measurement result
	InitialiseSampleMeasurement (see page 243)	Initialise the dialog controls with a sample measurement result
	PopulateListCtrl (see page 244)	Populate the list control with results and normals.
	UseBSMeasurement (see page 247)	Allows the caller to specify a downloaded measurement structure to be use when initialising the dialog. Must be called before the form is loaded (displayed)

4.3.1.1.2.1 FrmBodystatResults Fields

4.3.1.1.2.1.1 FrmBodystatResults.bsmMeasurement Field

C++

```
BodystatAPI.BodystatAPI.BSMeasurement bsmMeasurement;
```

C#

```
private BodystatAPI.BodystatAPI.BSMeasurement bsmMeasurement;
```

Visual Basic

```
Private bsmMeasurement As BodystatAPI.BodystatAPI.BSMeasurement
```

Java

```
private BodystatAPI.BodystatAPI.BSMeasurement bsmMeasurement;
```

MATLAB

```
bsmMeasurement
```

Description

Base the results dialog on this downloaded measurement

4.3.1.1.2.1.2 FrmBodystatResults.bUseActualMeasurment Field

C++

```
Boolean bUseActualMeasurment = False;
```

C#

```
private Boolean bUseActualMeasurment = False;
```

Visual Basic

```
Private bUseActualMeasurment As Boolean = False
```

Java

```
private Boolean bUseActualMeasurment = False;
```

MATLAB

```
bUseActualMeasurment
```

Description

Determines whether the dialog should use the measurement values or sample values

4.3.1.1.2.1.3 FrmBodystatResults.dtTestDate Field

C++

```
DateTime dtTestDate;
```

C#

```
private DateTime dtTestDate;
```

Visual Basic

```
Private dtTestDate As DateTime
```

Java

```
private DateTime dtTestDate;
```

MATLAB

```
dtTestDate
```

Description

Test date/time, not presently shown on the form (but may be passed to clipboard data)

4.3.1.1.2.2 FrmBodystatResults Methods

4.3.1.1.2.2.1 FrmBodystatResults.btnCalculateResultsButton_Click Method

Executes the Calculate Results button action. Uses the Bodystat API to calculate the results and normals for this test measurement.

C++

```
btnCalculateResultsButton_Click(System.Object sender, System.EventArgs e);
```

C#

```
private btnCalculateResultsButton_Click(System.Object sender, System.EventArgs e);
```

Visual Basic

```
Private Sub btnCalculateResultsButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCalculateresultsButton.Click
```

Java

```
private btnCalculateResultsButton_Click(System.Object sender, System.EventArgs e);
```

MATLAB

```
btnCalculateResultsButton_Click
```

Body Source

```
Private Sub btnCalculateResultsButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCalculateresultsButton.Click

    lstviewResultsList.Items.Clear()        ' Clear the results list

    If Me.cboModelCombo.SelectedIndex = BodystatAPI.BSDeviceModel.BSQuadScanTouch Or
Me.cboModelCombo.SelectedIndex = BodystatAPI.BSDeviceModel.BSMultiScanTouch Then
        MsgBox("The selected model is not supported by this SDK sample.",
MsgBoxStyle.Information)
        Exit Sub
    End If

    '
    ' Prepare input structure for the Bodystat SDK (using member values from form data)
    ' More likely this would be obtained directly from a BSMeasurement downloaded from the
unit
    '
    Dim bm As BodystatAPI.BodystatAPI.BSMeasurement
    bm.iDeviceModel = Me.cboModelCombo.SelectedIndex    ' Use our combo selection for
Bodystat model type (its a direct mapping)
    bm.tTestDate = Me.dtTestDate                        ' Not relevant for performing
calculations, but might be copied to clipboard later
    bm.iGender = IIf(Me.rdoGenderMaleRadio.Checked, BodystatAPI.BSGender.BSMale,
BodystatAPI.BSGender.BSFemale)        ' Convert our gender radio buttons into Bodystat
gender type
    bm.iAge = Me.txtAgeEdit.Text                        ' Age in years
    bm.iHeight = Me.txtHeightEdit.Text                  ' Height in centimeters
    bm.fWeight = Me.txtWeightEdit.Text                  ' Weight in kilograms
    bm.iActivity = Me.cboActivityCombo.SelectedIndex    ' Physical activity level (range 1
to 5) 1=very low 2=low/medium 3=medium 4=medium/high 5=very high
    bm.iWaist = Me.txtWaistEdit.Text                    ' Waist in centimeters
    bm.iHip = Me.txtHipEdit.Text                        ' Hip in centimeters
    bm.iZ_5kHz = Me.txtImp5kEdit.Text                   ' Impedance 5 KHz  (in ohms)
(MDD/Quad only)
    bm.iZ_50kHz = Me.txtImp50kEdit.Text                 ' Impedance 50 KHz  (in ohms)
    bm.iZ_100kHz = Me.txtImp100kEdit.Text               ' Impedance 100 KHz (in ohms) (Quad
only)
    bm.iZ_200kHz = Me.txtImp200kEdit.Text               ' Impedance 200 KHz (in ohms) (Quad
only)
    bm.iR_50kHz = Me.txtRes50kEdit.Text                 ' Resistance 50 kHz (in ohms)
(MDD/Quad only)
    bm.fX_50kHz = Me.txtReac50kEdit.Text                ' Reactance 50 kHz  (in ohms)
(MDD/Quad only)
    bm.fPA_50kHz = Me.txtPhaseAngle50kEdit.Text         ' Phase angle 50 kHz (in degrees)
(MDD/Quad only)
    bm.iFrequencies = 0                                 ' MST frequencies not supported by
this sample

    ' Choose analysis mode based upon selected analysis checkboxes
    ' These options might normally be stored as program setting rather than asked of the
user each time (like the installation mode of the device)
    ' The analysis mode impacts the equations used behind some results in certain scenarios
    Dim iAnalysisMode As BodystatAPI.BSAnalysisMode
    If (Me.chkModeBodyCompCheck.Checked And Me.chkModeHydrationCheck.Checked) Then
        iAnalysisMode = BodystatAPI.BSAnalysisMode.BSAnalysisModeBoth
    ElseIf (chkModeBodyCompCheck.Checked) Then
        iAnalysisMode = BodystatAPI.BSAnalysisMode.BSAnalysisModeBC
    ElseIf (chkModeHydrationCheck.Checked) Then
        iAnalysisMode = BodystatAPI.BSAnalysisMode.BSAnalysisModeHN
    Else
        iAnalysisMode = BodystatAPI.BSAnalysisMode.BSAnalysisModeNone
    End If

    '
    ' Get the results from the Bodystat SDK
    Dim bseStatus As BodystatAPI.BSError
    Dim br As BodystatAPI.BodystatAPI.BSResults
    bseStatus = BodystatAPI.BodystatAPI.BSCalculateResults(bm, br, iAnalysisMode)   '
Results stored in br structure

    ' Check for any errors from the calculation function
    If (bseStatus <> BodystatAPI.BSError.NoError) Then
        MsgBox(String.Format("Error calculating results." & vbCrLf & "Error Code: {0}",
bseStatus), MsgBoxStyle.Exclamation)
        Return         ' Nothing to display
    End If

    '
    ' Get the normal ranges from the Bodystat SDK
    '
    Dim bn As BodystatAPI.BodystatAPI.BSNormals
    bseStatus = BodystatAPI.BodystatAPI.BSCalculateNormals(bm, br, bn)    ' Normals stored
in bn structure

    ' Check for any errors from the normals function
    If (bseStatus <> BodystatAPI.BSError.NoError) Then
        MsgBox(String.Format("Error calculating normals." & vbCrLf & "Error Code: {0}",
bseStatus), MsgBoxStyle.Exclamation)
        Return      ' Nothing to display
    End If

    ' Populate the list control with the calculated results and calculated normals
    PopulateListCtrl(bm, br, bn)

End Sub
```

4.3.1.1.2.2.2 FrmBodystatResults.btnCopytoclipboardButton_Click Method

Implements the Copy To Clipboard button action. Copies the various parameters relating to a measurement directly to the clipboard

C++

```
btnCopytoclipboardButton_Click(System.Object sender, System.EventArgs e);
```

C#

```
private btnCopytoclipboardButton_Click(System.Object sender, System.EventArgs e);
```

Visual Basic

```
Private Sub btnCopytoclipboardButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCopytoclipboardButton.Click
```

Java

```
private btnCopytoclipboardButton_Click(System.Object sender, System.EventArgs e);
```

MATLAB

```
btnCopytoclipboardButton_Click
```

Body Source

```
Private Sub btnCopytoclipboardButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCopytoclipboardButton.Click

    Dim strBuffer As New StringBuilder()

    ' Output the measurement input parameters to a buffer ready for the clipboard
    strBuffer.Append("Bodystat Data" & vbCrLf)
    strBuffer.Append("-------------" & vbCrLf)

    Dim strModel As New StringBuilder(32)
    BodystatAPI.BodystatAPI.BSGetDeviceModelName(cboModelCombo.SelectedIndex, strModel,
strModel.Capacity)
    strBuffer.AppendFormat("Model:" & vbTab & "{0}" & vbCrLf, strModel.ToString)

    strBuffer.AppendFormat("Test Date/Time:" & vbTab & "{0}" & vbCrLf,
dtTestDate.ToString())
    strBuffer.AppendFormat("Gender:" & vbTab & "{0}" & vbCrLf,
IIf(rdoGenderMaleRadio.Checked, "Male", "Female"))
    strBuffer.AppendFormat("Age:" & vbTab & "{0}" & vbTab & "years" & vbCrLf,
txtAgeEdit.Text)
    strBuffer.AppendFormat("Height:" & vbTab & "{0}" & vbTab & "cm" & vbCrLf,
txtHeightEdit.Text)
    strBuffer.AppendFormat("Weight:" & vbTab & "{0:F1}" & vbTab & "kg" & vbCrLf,
txtWeightEdit.Text)
    strBuffer.AppendFormat("Activity Level:" & vbTab & "{0}" & vbCrLf,
cboActivityCombo.SelectedItem.ToString())
    strBuffer.AppendFormat("Waist:" & vbTab & "{0}" & vbTab & "cm" & vbCrLf,
txtWaistEdit.Text)
    strBuffer.AppendFormat("Hip:" & vbTab & "{0}" & vbTab & "cm" & vbCrLf, txtHipEdit.Text)

    strBuffer.AppendFormat("Body Composition Mode:" & vbTab & "{0}" & vbCrLf,
IIf(chkModeBodyCompCheck.Checked, "Enabled", "Disabled"))
    strBuffer.AppendFormat("Hydration Mode:" & vbTab & "{0}" & vbCrLf,
IIf(chkModeHydrationCheck.Checked, "Enabled", "Disabled"))

    strBuffer.AppendFormat("Impedance 5 kHz:" & vbTab & "{0}" & vbTab & "ohm" & vbCrLf,
txtImp5kEdit.Text)
    strBuffer.AppendFormat("Impedance 50 kHz:" & vbTab & "{0}" & vbTab & "ohm" & vbCrLf,
txtImp50kEdit.Text)
    strBuffer.AppendFormat("Impedance 100 kHz:" & vbTab & "{0}" & vbTab & "ohm" & vbCrLf,
txtImp100kEdit.Text)
    strBuffer.AppendFormat("Impedance 200 kHz:" & vbTab & "{0}" & vbTab & "ohm" & vbCrLf,
txtImp200kEdit.Text)
    strBuffer.AppendFormat("Resistance 50 kHz:" & vbTab & "{0}" & vbTab & "ohm" & vbCrLf,
txtRes50kEdit.Text)
    strBuffer.AppendFormat("Reactance 50 kHz:" & vbTab & "{0:F1}" & vbTab & "ohm" & vbCrLf,
txtReac50kEdit.Text)
    strBuffer.AppendFormat("Phase Angle 50 kHz:" & vbTab & "{0:F1}" & vbTab & "°" & vbCrLf,
txtPhaseAngle50kEdit.Text)

    strBuffer.AppendLine()
```

```
        ' Copy the buffer to the clipboard
        Clipboard.SetText(strBuffer.ToString())
    End Sub
```

4.3.1.1.2.2.3 FrmBodystatResults.FrmBodystatResults_Load Method

Initialise the dialog controls with their content

C++

```
FrmBodystatResults_Load(System.Object sender, System.EventArgs e);
```

C#

```
private FrmBodystatResults_Load(System.Object sender, System.EventArgs e);
```

Visual Basic

```
Private Sub FrmBodystatResults_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
```

Java

```
private FrmBodystatResults_Load(System.Object sender, System.EventArgs e);
```

MATLAB

```
FrmBodystatResults_Load
```

Body Source

```
Private Sub FrmBodystatResults_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        '
        ' Populate model combo with all possible models
        '
        Me.cboModelCombo.Items.Add("Unknown")
        Dim strModel As New StringBuilder(32)
        Dim strComboItem As String
        Dim bsErr As BodystatAPI.BSError = BodystatAPI.BSError.NoError
        Dim iModel As Integer = 1
        Do
            bsErr = BodystatAPI.BodystatAPI.BSGetDeviceModelName(iModel, strModel,
strModel.Capacity)
            If (bsErr = BodystatAPI.BSError.NoError) Then

                strComboItem = String.Format("{0} (ID:{1})", strModel.ToString(), iModel)      '
Append model id number to the string (for illustration and to make it unique for the combo)
                Me.cboModelCombo.Items.Add(strComboItem)

            End If
            iModel += 1
        Loop While (bsErr = BodystatAPI.BSError.NoError)      ' Stop when we run out of valid
model ids

        ' Setup the results list control with three columns ready for the results and normal
values
        lstviewResultsList.Columns.Add("Result", 250)
        lstviewResultsList.Columns.Add("Value", 110)
        lstviewResultsList.Columns.Add("Normal", 140)

        ' Populate the measurement control values based on the downloaded measurement or with
sample data as appropriate
        If (bUseActualMeasurment) Then
            InitialiseFromMeasurement()
        Else
            InitialiseSampleMeasurement()
        End If

End Sub
```

4.3.1.1.2.2.4 FrmBodystatResults.InitialiseFromMeasurement Method

Initialise the dialog controls with a values from an actual measurement result

C++

```
InitialiseFromMeasurement();
```

C#

```
public InitialiseFromMeasurement();
```

Visual Basic

```
Public Sub InitialiseFromMeasurement()
```

Java

```
public InitialiseFromMeasurement();
```

MATLAB

```
InitialiseFromMeasurement
```

Body Source

```
Public Sub InitialiseFromMeasurement()

    Me.cboModelCombo.SelectedIndex = bsmMeasurement.iDeviceModel
    'Me.dtTestDate = bsmMeasurement.tTestDate       ' Not shown on dialog presently
    Me.rdoGenderMaleRadio.Checked = bsmMeasurement.iGender = BodystatAPI.BSGender.BSMale
    Me.rdoGenderFemaleRadio.Checked = bsmMeasurement.iGender = BodystatAPI.BSGender.BSFemale
    Me.txtAgeEdit.Text = bsmMeasurement.iAge
    Me.txtHeightEdit.Text = bsmMeasurement.iHeight
    Me.txtWeightEdit.Text = bsmMeasurement.fWeight
    Me.cboActivityCombo.SelectedIndex = bsmMeasurement.iActivity
    Me.txtWaistEdit.Text = bsmMeasurement.iWaist
    Me.txtHipEdit.Text = bsmMeasurement.iHip

    Me.chkModeBodyCompCheck.Checked = True
    Me.chkModeHydrationCheck.Checked = True

    Me.txtImp5kEdit.Text = bsmMeasurement.iZ_5kHz
    Me.txtImp50kEdit.Text = bsmMeasurement.iZ_50kHz
    Me.txtImp100kEdit.Text = bsmMeasurement.iZ_100kHz
    Me.txtImp200kEdit.Text = bsmMeasurement.iZ_200kHz
    Me.txtRes50kEdit.Text = bsmMeasurement.iR_50kHz
    Me.txtReac50kEdit.Text = bsmMeasurement.fX_50kHz
    Me.txtPhaseAngle50kEdit.Text = bsmMeasurement.fPA_50kHz

End Sub
```

4.3.1.1.2.2.5 FrmBodystatResults.InitialiseSampleMeasurement Method

Initialise the dialog controls with a sample measurement result

C++

```
InitialiseSampleMeasurement();
```

C#

```
public InitialiseSampleMeasurement();
```

Visual Basic

```
Public Sub InitialiseSampleMeasurement()
```

Java

```
public InitialiseSampleMeasurement();
```

MATLAB

```
InitialiseSampleMeasurement
```

Body Source

```
Public Sub InitialiseSampleMeasurement()

    ' Setup the form with some sensible default values
    Me.cboModelCombo.SelectedIndex = BodystatAPI.BSDeviceModel.BSQuadScan_BT
```

```
        'Me.dtTestDate = DateTime.Now()       ' Not shown on dialog presently
        Me.rdoGenderMaleRadio.Checked = True
        Me.rdoGenderFemaleRadio.Checked = False
        Me.txtAgeEdit.Text = "30"
        Me.txtHeightEdit.Text = "165"
        Me.txtWeightEdit.Text = "85.0"
        Me.cboActivityCombo.SelectedIndex = 3
        Me.txtWaistEdit.Text = "85"
        Me.txtHipEdit.Text = "90"

        Me.chkModeBodyCompCheck.Checked = True
        Me.chkModeHydrationCheck.Checked = True

        Me.txtImp5kEdit.Text = "500"
        Me.txtImp50kEdit.Text = "500"
        Me.txtImp100kEdit.Text = "500"
        Me.txtImp200kEdit.Text = "500"
        Me.txtRes50kEdit.Text = "500"
        Me.txtReac50kEdit.Text = "52.0"
        Me.txtPhaseAngle50kEdit.Text = "6.1"
    End Sub
```

4.3.1.1.2.2.6 FrmBodystatResults.PopulateListCtrl Method

Populate the list control with results and normals.

C++

```
PopulateListCtrl(BodystatAPI.BodystatAPI.BSMeasurement ^ bm,
BodystatAPI.BodystatAPI.BSResults ^ br, BodystatAPI.BodystatAPI.BSNormals ^ bn);
```

C#

```
public PopulateListCtrl(ref BodystatAPI.BodystatAPI.BSMeasurement bm, ref
BodystatAPI.BodystatAPI.BSResults br, ref BodystatAPI.BodystatAPI.BSNormals bn);
```

Visual Basic

```
Public Sub PopulateListCtrl(ByRef bm As BodystatAPI.BodystatAPI.BSMeasurement, ByRef br As
BodystatAPI.BodystatAPI.BSResults, ByRef bn As BodystatAPI.BodystatAPI.BSNormals)
```

Java

```
public PopulateListCtrl(BodystatAPI.BodystatAPI.BSMeasurement bm,
BodystatAPI.BodystatAPI.BSResults br, BodystatAPI.BodystatAPI.BSNormals bn);
```

MATLAB

```
PopulateListCtrl
```

Parameters

Parameters	Description
ByRef bm As BodystatAPI.BodystatAPI.BSMeasurement	The measurement parameters for this test.
ByRef br As BodystatAPI.BodystatAPI.BSResults	The calculated results for this test from the SDK.
ByRef bn As BodystatAPI.BodystatAPI.BSNormals	The calculated normals for this test from the SDK.

Body Source

```
Public Sub PopulateListCtrl(ByRef bm As BodystatAPI.BodystatAPI.BSMeasurement, ByRef br As
BodystatAPI.BodystatAPI.BSResults, ByRef bn As BodystatAPI.BodystatAPI.BSNormals)

    Dim lvi As ListViewItem

    ' Fat %
    lvi = lstviewResultsList.Items.Add("Fat percentage")
    lvi.SubItems.Add(String.Format("{0:F1}", br.fFatPerc))
    lvi.SubItems.Add(String.Format("{0} - {1}", bn.iFatPerc_L, bn.iFatPerc_H))

    ' Fat kg
    lvi = lstviewResultsList.Items.Add("Fat (in kg)")
    lvi.SubItems.Add(String.Format("{0:F1}", br.fFatKg))
    lvi.SubItems.Add(String.Format("{0} - {1}", bn.iFatKg_L, bn.iFatKg_H))
    ' Lean %
    lvi = lstviewResultsList.Items.Add("Lean percentage")
    lvi.SubItems.Add(String.Format("{0:F1}", br.fLeanPerc))
    lvi.SubItems.Add(String.Format("{0} - {1}", bn.iLeanPerc_L, bn.iLeanPerc_H))

    ' Lean kg
    lvi = lstviewResultsList.Items.Add("Lean (in kg)")
    lvi.SubItems.Add(String.Format("{0:F1}", br.fLeanKg))
    lvi.SubItems.Add(String.Format("{0} - {1}", bn.iLeanKg_L, bn.iLeanKg_H))

    ' Total Weight
    lvi = lstviewResultsList.Items.Add("Total Weight (in kg)")
    lvi.SubItems.Add(String.Format("{0:F1}", br.fTotalWeight))
    ' Total weight normal range is calculated using composition or BMI (whichever is the
greater). We indicate which method was used in brackets after the value.
    lvi.SubItems.Add(String.Format("{0} - {1} ({2})", bn.iTotalWeight_L, bn.iTotalWeight_H,
IIf(bn.iTotalWeightMethod = 0, "Composition", "BMI")))

    ' Dry lean weight
    lvi = lstviewResultsList.Items.Add("Dry Lean Weight (in kg)")
    lvi.SubItems.Add(String.Format("{0:F1}", br.fDryLW))
    lvi.SubItems.Add("N/A")

    ' TBW %
    lvi = lstviewResultsList.Items.Add("Total Body Water percentage")
    lvi.SubItems.Add(String.Format("{0:F1}", br.fTBWPerc))
    lvi.SubItems.Add(String.Format("{0} - {1}", bn.iTBWPerc_L, bn.iTBWPerc_H))

    ' TBW lt
    lvi = lstviewResultsList.Items.Add("Total Body Water (in litres)")
    lvi.SubItems.Add(String.Format("{0:F1}", br.fTBW))
    lvi.SubItems.Add(String.Format("{0} - {1}", bn.iTBW_L, bn.iTBW_H))

    ' ECW %
    lvi = lstviewResultsList.Items.Add("Extra Cellular Water percentage")
    lvi.SubItems.Add(IIf(BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) =
BodystatAPI.BSDeviceFamily.BSQuadScan_Family Or
BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) =
BodystatAPI.BSDeviceFamily.BSMDD_Family, String.Format("{0:F1}", br.fECWPerc), "(MDD/Quad
Only)"))
    lvi.SubItems.Add(IIf(BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) =
BodystatAPI.BSDeviceFamily.BSQuadScan_Family Or
BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) =
BodystatAPI.BSDeviceFamily.BSMDD_Family, String.Format("{0}", bn.iECWPerc_Norm), "(MDD/Quad
Only)"))

    ' ECW lt
    lvi = lstviewResultsList.Items.Add("Extra Cellular Water (in litres)")
    lvi.SubItems.Add(IIf(BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) =
BodystatAPI.BSDeviceFamily.BSQuadScan_Family Or
BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) =
BodystatAPI.BSDeviceFamily.BSMDD_Family, String.Format("{0:F1}", br.fECW), "(MDD/Quad
Only)"))
    lvi.SubItems.Add("N/A")

    ' ICW %
    lvi = lstviewResultsList.Items.Add("Intra Cellular Water percentage")
    lvi.SubItems.Add(IIf(BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) =
BodystatAPI.BSDeviceFamily.BSQuadScan_Family Or
BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) =
BodystatAPI.BSDeviceFamily.BSMDD_Family, String.Format("{0:F1}", br.fICWPerc), "(MDD/Quad
Only)"))
    lvi.SubItems.Add(IIf(BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) =
BodystatAPI.BSDeviceFamily.BSQuadScan_Family Or
BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) =
BodystatAPI.BSDeviceFamily.BSMDD_Family, String.Format("{0}", bn.iICWPerc_Norm), "(MDD/Quad
Only)"))

    ' ICW lt
    lvi = lstviewResultsList.Items.Add("Intra Cellular Water (in litres)")
    lvi.SubItems.Add(IIf(BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) =
BodystatAPI.BSDeviceFamily.BSQuadScan_Family Or
BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) =
BodystatAPI.BSDeviceFamily.BSMDD_Family, String.Format("{0:F1}", br.fICW), "(MDD/Quad
Only)"))
    lvi.SubItems.Add("N/A")

    ' Body Cell Mass
    lvi = lstviewResultsList.Items.Add("Body Cell Mass")
    lvi.SubItems.Add(IIf(BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) =
BodystatAPI.BSDeviceFamily.BSQuadScan_Family, String.Format("{0:F1}", br.fBCM), "(Quad
Only)"))
    lvi.SubItems.Add("N/A")

    ' 3rd space water
    lvi = lstviewResultsList.Items.Add("3rd space water (in litres)")
    lvi.SubItems.Add(IIf(BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) =
BodystatAPI.BSDeviceFamily.BSQuadScan_Family, String.Format("{0:F1}", br.fThirdSpace),
"(Quad Only)"))
    lvi.SubItems.Add("N/A")

    ' Nutrition index
    lvi = lstviewResultsList.Items.Add("Nutrition index")
    lvi.SubItems.Add(IIf(BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) =
BodystatAPI.BSDeviceFamily.BSQuadScan_Family, String.Format("{0:F2}", br.fNutrition),
"(Quad Only)"))
    lvi.SubItems.Add(IIf(BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) =
BodystatAPI.BSDeviceFamily.BSQuadScan_Family, String.Format("{0:F2}", bn.fNutrition_Norm),
"(Quad Only)"))

    ' Prediction marker - formerly known as Illness Marker (TM)
    lvi = lstviewResultsList.Items.Add("Prediction Marker (TM)")
    lvi.SubItems.Add(IIf(BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) =
BodystatAPI.BSDeviceFamily.BSQuadScan_Family, String.Format("{0:F3}", br.fIllness), "(Quad
Only)"))
    lvi.SubItems.Add(IIf(BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) =
BodystatAPI.BSDeviceFamily.BSQuadScan_Family, String.Format("{0:F3} - {1:F3}",
bn.fIllness_L, bn.fIllness_H), "(Quad Only)"))

    ' BMR kcal
    lvi = lstviewResultsList.Items.Add("Basal Metabolic Rate (in kcal)")
    lvi.SubItems.Add(String.Format("{0:F0}", br.fBMR))
    lvi.SubItems.Add("N/A")

    ' BMR kcal/kg
    lvi = lstviewResultsList.Items.Add("Basal Metabolic Rate per kilogram (in kcal/kg)")
    lvi.SubItems.Add(String.Format("{0:F1}", br.fBMRkg))
    lvi.SubItems.Add("N/A")

    ' EAR
    lvi = lstviewResultsList.Items.Add("Estimated Average Requirement (in kcal)")
    lvi.SubItems.Add(String.Format("{0:F0}", br.fEstAvg))
    lvi.SubItems.Add("N/A")

    ' BMI
    lvi = lstviewResultsList.Items.Add("Body Mass Index (BMI)")
    lvi.SubItems.Add(String.Format("{0:F1}", br.fBMI))
    lvi.SubItems.Add(String.Format("{0} - {1}", bn.iBMI_L, bn.iBMI_H))

    ' BFMI
    lvi = lstviewResultsList.Items.Add("Body Fat Mass Index (BFMI)")
    lvi.SubItems.Add(IIf(BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) =
BodystatAPI.BSDeviceFamily.BSQuadScan_Family Or
BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) =
BodystatAPI.BSDeviceFamily.BSMDD_Family, String.Format("{0:F1}", br.fBFMI), "(MDD/Quad
Only)"))
    lvi.SubItems.Add(IIf(BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) =
BodystatAPI.BSDeviceFamily.BSQuadScan_Family Or
BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) =
BodystatAPI.BSDeviceFamily.BSMDD_Family, String.Format("{0} - {1}", bn.iBFMI_L,
bn.iBFMI_H), "(MDD/Quad Only)"))

    ' FFMI
    lvi = lstviewResultsList.Items.Add("Fat Free Mass Index (FFMI)")
    lvi.SubItems.Add(IIf(BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) =
BodystatAPI.BSDeviceFamily.BSQuadScan_Family Or
BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) =
BodystatAPI.BSDeviceFamily.BSMDD_Family, String.Format("{0:F1}", br.fFFMI), "(MDD/Quad
Only)"))
    lvi.SubItems.Add(IIf(BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) =
BodystatAPI.BSDeviceFamily.BSQuadScan_Family Or
BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) =
BodystatAPI.BSDeviceFamily.BSMDD_Family, String.Format("{0} - {1}", bn.iFFMI_L,
bn.iFFMI_H), "(MDD/Quad Only)"))

    ' Waist/Hip
    lvi = lstviewResultsList.Items.Add("Waist/Hip ratio")
    lvi.SubItems.Add(String.Format("{0:F2}", br.fWaistHip))
    lvi.SubItems.Add(String.Format("{0:F2}", bn.fWaistHip_Norm))

    ' Wellness marker
    lvi = lstviewResultsList.Items.Add("Wellness Marker (TM)")
    lvi.SubItems.Add(IIf(BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) =
BodystatAPI.BSDeviceFamily.BSMDD_Family, String.Format("{0:F3}", br.fWellness), "(MDD
Only)"))
    lvi.SubItems.Add(IIf(BodystatAPI.BodystatAPI.BSGetDeviceFamily(bm.iDeviceModel) =
BodystatAPI.BSDeviceFamily.BSMDD_Family, String.Format("{0:F3} - {0:F3}", bn.fWellness_L,
bn.fWellness_H), "(MDD Only)"))
End Sub
```

4.3.1.1.2.2.7 FrmBodystatResults.UseBSMeasurement Method

Allows the caller to specify a downloaded measurement structure to be use when initialising the dialog. Must be called before the form is loaded (displayed)

C++

```
UseBSMeasurement(BodystatAPI.BodystatAPI.BSMeasurement bsm);
```

C#

```
public UseBSMeasurement(BodystatAPI.BodystatAPI.BSMeasurement bsm);
```

Visual Basic

```
Public Sub UseBSMeasurement(ByVal bsm As BodystatAPI.BodystatAPI.BSMeasurement)
```

Java

```
public UseBSMeasurement(BodystatAPI.BodystatAPI.BSMeasurement bsm);
```

MATLAB

```
UseBSMeasurement
```

Body Source

```
Public Sub UseBSMeasurement(ByVal bsm As BodystatAPI.BodystatAPI.BSMeasurement)
    bsmMeasurement = bsm
    bUseActualMeasurment = True
End Sub
```

Index

B

Bodystat 2

Bodystat API 1, 2, 57

- Functions 6
- Structs, Records, Enums 29
- Variables 52

Bodystat API - Main 1

- Files 54
- Macros 53

Bodystat C# WinForms Sample 204

- Files 215

Bodystat MFC Sample 155

- Classes 155

Bodystat Visual Basic .NET Sample 216

Bodystat::BS_RAWDATA_ARRAYSIZE 52

Bodystat::BS_RAWDATA_ARRAYSIZE variable 52

Bodystat::BSAnalysisMode 30

Bodystat::BSAnalysisMode enumeration 30

Bodystat::BSAuthenticateBTDevice 8

Bodystat::BSAuthenticateBTDevice function 8

Bodystat::BSAutoSetupBT 8

Bodystat::BSAutoSetupBT function 8

Bodystat::BSCalculateNormals 10

Bodystat::BSCalculateNormals function 10

Bodystat::BSCalculateResults 10

Bodystat::BSCalculateResults function 10

Bodystat::BSCloseComport 11

Bodystat::BSCloseComport function 11

Bodystat::BSConnect 11

Bodystat::BSConnect function 11

Bodystat::BSDeviceFamily 30

Bodystat::BSDeviceFamily enumeration 30

Bodystat::BSDeviceModel 32

Bodystat::BSDeviceModel enumeration 32

Bodystat::BSError 33

Bodystat::BSError enumeration 33

Bodystat::BSGender 35

Bodystat::BSGender enumeration 35

Bodystat::BSGetBTBodystatDevice 12

Bodystat::BSGetBTBodystatDevice function 12

Bodystat::BSGetBTStackInfo 13

Bodystat::BSGetBTStackInfo function 13

Bodystat::BSGetDeviceFamily 13

Bodystat::BSGetDeviceFamily function 13

Bodystat::BSGetDeviceModelName 14

Bodystat::BSGetDeviceModelName function 14

Bodystat::BSGetDeviceModelNameShort 14

Bodystat::BSGetDeviceModelNameShort function 14

Bodystat::BSGetSdkLibraryVersion 15

Bodystat::BSGetSdkLibraryVersion function 15

Bodystat::BSIsBTAvailable 16

Bodystat::BSIsBTAvailable function 16

Bodystat::BSMeasurement 36

Bodystat::BSMeasurement structure 36

Bodystat::BSNormals 38

Bodystat::BSNormals structure 38

Bodystat::BSOpenComport 16

Bodystat::BSOpenComport function 16

Bodystat::BSRawData 40

Bodystat::BSRawData structure 40

Bodystat::BSReadCalibrationTime 17

Bodystat::BSReadCalibrationTime function 17

Bodystat::BSReadCurrentTime 17

Bodystat::BSReadCurrentTime function 17

Bodystat::BSReadModelVersion 18

Bodystat::BSReadModelVersion function 18

Bodystat::BSReadPrinterAddress 19

Bodystat::BSReadPrinterAddress function 19

Bodystat::BSReadProtocolInfo 19

Bodystat::BSReadProtocolInfo function 19

Bodystat::BSReadSerialNumber 20

Bodystat::BSReadSerialNumber function 20

Bodystat::BSReadStoredTestData 21

Bodystat::BSReadStoredTestData function 21

Bodystat::BSReadTestBodystat 21

Bodystat::BSReadTestBodystat function 21

Bodystat::BSResults 41

Bodystat::BSResults structure 41

Bodystat::BSSearchBTDevices 22

Bodystat::BSSearchBTDevices function 22

Bodystat::BSUnAuthenticateBTDevices 23

Bodystat::BSUnAuthenticateBTDevices function 23
Bodystat::BSWriteCurrentTime 24
Bodystat::BSWriteCurrentTime function 24
Bodystat::BSWriteCurrentTimeAs 24
Bodystat::BSWriteCurrentTimeAs function 24
Bodystat::BSWritePrinterAddress 25
Bodystat::BSWritePrinterAddress function 25
Bodystat::ES_RAWDATA_ARRAYSIZE 52
Bodystat::ES_RAWDATA_ARRAYSIZE variable 52
Bodystat::ESCalculateNormals 26
Bodystat::ESCalculateNormals function 26
Bodystat::ESCalculateResults 26
Bodystat::ESCalculateResults function 26
Bodystat::ESMeasurement 44
Bodystat::ESMeasurement structure 44
Bodystat::ESNormals 45
Bodystat::ESNormals structure 45
Bodystat::ESRawData 47
Bodystat::ESRawData structure 47
Bodystat::ESReadStoredTestData 27
Bodystat::ESReadStoredTestData function 27
Bodystat::ESReadTestEquistat 28
Bodystat::ESReadTestEquistat function 28
Bodystat::ESResults 48
Bodystat::ESResults structure 48
Bodystat::ImpData 49
Bodystat::ImpData::fFrequency 50
Bodystat::ImpData::fImpedance 50
Bodystat::ImpData::fPhaseAngle 51
Bodystat::ImpData::ImZ 51
Bodystat::ImpData::ReZ 51
BodystatAPI 57
BodystatAPI class 81
  about BodystatAPI class 81
  BS_RAWDATA_ARRAYSIZE 114
  BSAuthenticateBTDevice 115
  BSAutoSetupBT 116
  BSCalculateNormals 117
  BSCalculateResults 118
  BSCloseComport 119
  BSConnect 120
  BSGetBTBodystatDevice 120
  BSGetBTStackInfo 121
  BSGetDeviceFamily 122
  BSGetDeviceModelName 123
  BSGetDeviceModelNameShort 124
  BSGetSdkLibraryVersion 124
  BSIsBTAvailable 125
  BSOpenComport 126
  BSReadCalibrationTime 127
  BSReadCurrentTime 128, 129
  BSReadModelVersion 130
  BSReadPrinterAddress 131
  BSReadProtocolInfo 132
  BSReadSerialNumber 133
  BSReadStoredTestData 133
  BSReadTestBodystat 134
  BSSearchBTDevices 136
  BSUnAuthenticateBTDevices 137
  BSWriteCurrentTime 138
  BSWriteCurrentTimeAs 138, 139
  BSWritePrinterAddress 140
  ES_RAWDATA_ARRAYSIZE 115
  ESCalculateNormals 141
  ESCalculateResults 142
  ESReadStoredTestData 142
  ESReadTestEquistat 143
BodystatAPI.BodystatAPI 81
BodystatAPI.BodystatAPI.BS_RAWDATA_ARRAYSIZE 114
BodystatAPI.BodystatAPI.BSAuthenticateBTDevice 115
BodystatAPI.BodystatAPI.BSAutoSetupBT 116
BodystatAPI.BodystatAPI.BSAutoSetupBT@out string@bool@IntPtr 57
BodystatAPI.BodystatAPI.BSAutoSetupBT@StringBuilder@int@bool@IntPtr 58
BodystatAPI.BodystatAPI.BSCalculateNormals 117
BodystatAPI.BodystatAPI.BSCalculateNormals@BSMeasurement@BSResults@out BSNormals 59
BodystatAPI.BodystatAPI.BSCalculateResults 118
BodystatAPI.BodystatAPI.BSCalculateResults@BSMeasurement@out BSResults@BSAnalysisMode 59
BodystatAPI.BodystatAPI.BSCloseComport 60, 119
BodystatAPI.BodystatAPI.BSConnect 60, 120
BodystatAPI.BodystatAPI.BSGetBTBodystatDevice 120

BodystatAPI.BodystatAPI.BSGetBTBodystatDevice@out string@out string@out string@UInt16 77

BodystatAPI.BodystatAPI.BSGetBTBodystatDevice@StringBuilder@int@StringBuilder@int@StringBuilder@int@UInt16 78

BodystatAPI.BodystatAPI.BSGetBTStackInfo 121

BodystatAPI.BodystatAPI.BSGetBTStackInfo@out string 60

BodystatAPI.BodystatAPI.BSGetBTStackInfo@StringBuilder@int 60

BodystatAPI.BodystatAPI.BSGetDeviceFamily 122

BodystatAPI.BodystatAPI.BSGetDeviceModelName 123

BodystatAPI.BodystatAPI.BSGetDeviceModelName@BSDeviceModel@out string 60

BodystatAPI.BodystatAPI.BSGetDeviceModelName@BSDeviceModel@StringBuilder@int 61

BodystatAPI.BodystatAPI.BSGetDeviceModelNameShort 124

BodystatAPI.BodystatAPI.BSGetDeviceModelNameShort@BSDeviceModel@out string 61

BodystatAPI.BodystatAPI.BSGetDeviceModelNameShort@BSDeviceModel@StringBuilder@int 61

BodystatAPI.BodystatAPI.BSGetSdkLibraryVersion 124

BodystatAPI.BodystatAPI.BSGetSdkLibraryVersion@out int@out int 62

BodystatAPI.BodystatAPI.BSIsBTAvailable 62, 125

BodystatAPI.BodystatAPI.BSMeasurement 84

BodystatAPI.BodystatAPI.BSMeasurement.fPA_50kHz 86

BodystatAPI.BodystatAPI.BSMeasurement.fWeight 87

BodystatAPI.BodystatAPI.BSMeasurement.fX_50kHz 87

BodystatAPI.BodystatAPI.BSMeasurement.iActivity 87

BodystatAPI.BodystatAPI.BSMeasurement.iAge 88

BodystatAPI.BodystatAPI.BSMeasurement.iDeviceModel 88

BodystatAPI.BodystatAPI.BSMeasurement.iFrequencies 88

BodystatAPI.BodystatAPI.BSMeasurement.iGender 89

BodystatAPI.BodystatAPI.BSMeasurement.iHeight 89

BodystatAPI.BodystatAPI.BSMeasurement.iHip 89

BodystatAPI.BodystatAPI.BSMeasurement.iR_50kHz 90

BodystatAPI.BodystatAPI.BSMeasurement.iWaist 90

BodystatAPI.BodystatAPI.BSMeasurement.iZ_100kHz 90

BodystatAPI.BodystatAPI.BSMeasurement.iZ_200kHz 91

BodystatAPI.BodystatAPI.BSMeasurement.iZ_50kHz 91

BodystatAPI.BodystatAPI.BSMeasurement.iZ_5kHz 91

BodystatAPI.BodystatAPI.BSMeasurement.llTestDate 92

BodystatAPI.BodystatAPI.BSMeasurement.pMultifreqData 92

BodystatAPI.BodystatAPI.BSMeasurement.tTestDate 93

BodystatAPI.BodystatAPI.BSMeasurement.ulDeviceSerialNumber 92

BodystatAPI.BodystatAPI.BSNormals 93

BodystatAPI.BodystatAPI.BSNormals structure 93

BodystatAPI.BodystatAPI.BSNormals.AerobicCapacity_H 62

BodystatAPI.BodystatAPI.BSNormals.AerobicCapacity_L 62

BodystatAPI.BodystatAPI.BSNormals.BFMI_H 63

BodystatAPI.BodystatAPI.BSNormals.BFMI_L 63

BodystatAPI.BodystatAPI.BSNormals.BMI_H 63

BodystatAPI.BodystatAPI.BSNormals.BMI_L 63

BodystatAPI.BodystatAPI.BSNormals.Cholesterol_HDL_H 63

BodystatAPI.BodystatAPI.BSNormals.Cholesterol_HDL_L 63

BodystatAPI.BodystatAPI.BSNormals.Cholesterol_LDL_H 63

BodystatAPI.BodystatAPI.BSNormals.Cholesterol_LDL_L 63

BodystatAPI.BodystatAPI.BSNormals.Cholesterol_Total_H 63

BodystatAPI.BodystatAPI.BSNormals.Cholesterol_Total_L 63

BodystatAPI.BodystatAPI.BSNormals.Cholesterol_Triglycerides_H 64

BodystatAPI.BodystatAPI.BSNormals.Cholesterol_Triglycerides_L 64

BodystatAPI.BodystatAPI.BSNormals.Cholesterol_VLDL_H 64

BodystatAPI.BodystatAPI.BSNormals.Cholesterol_VLDL_L 64

BodystatAPI.BodystatAPI.BSNormals.ECWPerc_Norm 64

BodystatAPI.BodystatAPI.BSNormals.FatKg_H 64

BodystatAPI.BodystatAPI.BSNormals.FatKg_L 64

BodystatAPI.BodystatAPI.BSNormals.FatPerc_H 64

BodystatAPI.BodystatAPI.BSNormals.FatPerc_L 64

BodystatAPI.BodystatAPI.BSNormals.FFMI_H 65

BodystatAPI.BodystatAPI.BSNormals.FFMI_L 65

BodystatAPI.BodystatAPI.BSNormals.Flexibility_H 65

BodystatAPI.BodystatAPI.BSNormals.Flexibility_L 65

BodystatAPI.BodystatAPI.BSNormals.GlucoseFasting_H 65

BodystatAPI.BodystatAPI.BSNormals.GlucoseFasting_L 65

BodystatAPI.BodystatAPI.BSNormals.GlucoseNonFasting_H 65

BodystatAPI.BodystatAPI.BSNormals.GlucoseNonFasting_L 65

BodystatAPI.BodystatAPI.BSNormals.GripLeft 65

BodystatAPI.BodystatAPI.BSNormals.GripRight 65

BodystatAPI.BodystatAPI.BSNormals.ICWPerc_Norm 66

BodystatAPI.BodystatAPI.BSNormals.Illness_H 66

BodystatAPI.BodystatAPI.BSNormals.Illness_L 66

BodystatAPI.BodystatAPI.BSNormals.LeanKg_H 66

BodystatAPI.BodystatAPI.BSNormals.LeanKg_L 66

BodystatAPI.BodystatAPI.BSNormals.LeanPerc_H 66

BodystatAPI.BodystatAPI.BSNormals.LeanPerc_L 66

BodystatAPI.BodystatAPI.BSNormals.Nutrition_Norm 66

BodystatAPI.BodystatAPI.BSNormals.TBW_H 66

BodystatAPI.BodystatAPI.BSNormals.TBW_L 66

BodystatAPI.BodystatAPI.BSNormals.TBWPerc_H 66

BodystatAPI.BodystatAPI.BSNormals.TBWPerc_L 67

BodystatAPI.BodystatAPI.BSNormals.TotalWeight_H 67

BodystatAPI.BodystatAPI.BSNormals.TotalWeight_L 67

BodystatAPI.BodystatAPI.BSNormals.TotalWeightMethod 67

BodystatAPI.BodystatAPI.BSNormals.WaistHip_Norm 67

BodystatAPI.BodystatAPI.BSNormals.Wellness_H 67

BodystatAPI.BodystatAPI.BSNormals.Wellness_L 67

BodystatAPI.BodystatAPI.BSOpenComport 126

BodystatAPI.BodystatAPI.BSOpenComport@string 67

BodystatAPI.BodystatAPI.BSOpenComport@StringBuilder@int
68

BodystatAPI.BodystatAPI.BSRawData 96

BodystatAPI.BodystatAPI.BSRawData.BSRawData@int 68

BodystatAPI.BodystatAPI.BSRawData.iRecordArraySize 97

BodystatAPI.BodystatAPI.BSRawData.iTotalNumRecs 98

BodystatAPI.BodystatAPI.BSRawData.New 97

BodystatAPI.BodystatAPI.BSRawData.record 98

BodystatAPI.BodystatAPI.BSRawData.ulFirstTestNum 99

BodystatAPI.BodystatAPI.BSReadCalibrationTime 127

BodystatAPI.BodystatAPI.BSReadCalibrationTime@out DateTime 68

BodystatAPI.BodystatAPI.BSReadCalibrationTime@out Int64 68

BodystatAPI.BodystatAPI.BSReadCurrentTime 128, 129

BodystatAPI.BodystatAPI.BSReadCurrentTime@out DateTime@int 69

BodystatAPI.BodystatAPI.BSReadCurrentTime@out Int64@int 69

BodystatAPI.BodystatAPI.BSReadModelVersion 130

BodystatAPI.BodystatAPI.BSReadModelVersion@out BSDeviceModel@out byte@out byte@out byte@out byte@out byte 70

BodystatAPI.BodystatAPI.BSReadPrinterAddress 131

BodystatAPI.BodystatAPI.BSReadPrinterAddress@out string 70

BodystatAPI.BodystatAPI.BSReadPrinterAddress@StringBuilder@int
70

BodystatAPI.BodystatAPI.BSReadProtocolInfo 132

BodystatAPI.BodystatAPI.BSReadProtocolInfo@out byte@out byte@out byte 71

BodystatAPI.BodystatAPI.BSReadSerialNumber 133

BodystatAPI.BodystatAPI.BSReadSerialNumber@out uint 71

BodystatAPI.BodystatAPI.BSReadStoredTestData 133

BodystatAPI.BodystatAPI.BSReadTestBodystat 134

BodystatAPI.BodystatAPI.BSReadTestBodystat@out int@out int@out int@out int@out int@out float@out float 71

BodystatAPI.BodystatAPI.BSResults 99

BodystatAPI.BodystatAPI.BSResults structure 99

BodystatAPI.BodystatAPI.BSResults.BCM 72

BodystatAPI.BodystatAPI.BSResults.BFMI 72

BodystatAPI.BodystatAPI.BSResults.BMI 72

BodystatAPI.BodystatAPI.BSResults.BMR 72

BodystatAPI.BodystatAPI.BSResults.BMRkg 72

BodystatAPI.BodystatAPI.BSResults.DryLW 72

BodystatAPI.BodystatAPI.BSResults.ECW 73

BodystatAPI.BodystatAPI.BSResults.ECWPerc 73

BodystatAPI.BodystatAPI.BSResults.EstAvg 73

BodystatAPI.BodystatAPI.BSResults.FatKg 73

BodystatAPI.BodystatAPI.BSResults.FatPerc 73

BodystatAPI.BodystatAPI.BSResults.FFMI 73

BodystatAPI.BodystatAPI.BSResults.ICW 73

BodystatAPI.BodystatAPI.BSResults.ICWPerc 73

BodystatAPI.BodystatAPI.BSResults.Illness 73

BodystatAPI.BodystatAPI.BSResults.LeanKg 73

BodystatAPI.BodystatAPI.BSResults.LeanPerc 73

BodystatAPI.BodystatAPI.BSResults.Nutrition 74

BodystatAPI.BodystatAPI.BSResults.TBW 74

BodystatAPI.BodystatAPI.BSResults.TBWPerc 74

BodystatAPI.BodystatAPI.BSResults.ThirdSpace 74

BodystatAPI.BodystatAPI.BSResults.TotalWeight 74

BodystatAPI.BodystatAPI.BSResults.WaistHip 74

BodystatAPI.BodystatAPI.BSResults.Wellness 74

BodystatAPI.BodystatAPI.BSSearchBTDevices 136

BodystatAPI.BodystatAPI.BSSearchBTDevices@out int@out int@bool@IntPtr@UInt16 78

BodystatAPI.BodystatAPI.BSUnAuthenticateBTDevices 137

BodystatAPI.BodystatAPI.BSUnAuthenticateBTDevices@bool@IntPtr@UInt16
79

BodystatAPI.BodystatAPI.BSWriteCurrentTime 74, 138

BodystatAPI.BodystatAPI.BSWriteCurrentTimeAs 138, 139

BodystatAPI.BodystatAPI.BSWritePrinterAddress 140

BodystatAPI.BodystatAPI.BSWritePrinterAddress@string@int 74

BodystatAPI.BodystatAPI.BSWritePrinterAddress@StringBuilder@int 75

BodystatAPI.BodystatAPI.ES_RAWDATA_ARRAYSIZE 115

BodystatAPI.BodystatAPI.ESCalculateNormals 141

BodystatAPI.BodystatAPI.ESCalculateNormals@ESMeasurement@ESResults@out ESNormals 75

BodystatAPI.BodystatAPI.ESCalculateResults 142

BodystatAPI.BodystatAPI.ESCalculateResults@ESMeasurement@out ESResults 76

BodystatAPI.BodystatAPI.ESMeasurement 101

BodystatAPI.BodystatAPI.ESMeasurement.fPA_50kHz 103

BodystatAPI.BodystatAPI.ESMeasurement.iBCScore 103

BodystatAPI.BodystatAPI.ESMeasurement.iDeviceModel 104

BodystatAPI.BodystatAPI.ESMeasurement.iHeight 104

BodystatAPI.BodystatAPI.ESMeasurement.iZ_140kHz 104

BodystatAPI.BodystatAPI.ESMeasurement.iZ_16kHz 105

BodystatAPI.BodystatAPI.ESMeasurement.iZ_200kHz 105

BodystatAPI.BodystatAPI.ESMeasurement.iZ_24kHz 105

BodystatAPI.BodystatAPI.ESMeasurement.iZ_280kHz 106

BodystatAPI.BodystatAPI.ESMeasurement.iZ_50kHz 106

BodystatAPI.BodystatAPI.ESMeasurement.iZ_5kHz 106

BodystatAPI.BodystatAPI.ESMeasurement.llTestDate 107

BodystatAPI.BodystatAPI.ESMeasurement.tTestDate 107

BodystatAPI.BodystatAPI.ESMeasurement.ulDeviceSerialNumber 107

BodystatAPI.BodystatAPI.ESNormals 108

BodystatAPI.BodystatAPI.ESNormals structure 108

BodystatAPI.BodystatAPI.ESRawData 109

BodystatAPI.BodystatAPI.ESRawData.ESRawData@int 76

BodystatAPI.BodystatAPI.ESRawData.iRecordArraySize 111

BodystatAPI.BodystatAPI.ESRawData.iTotalNumRecs 111

BodystatAPI.BodystatAPI.ESRawData.New 110

BodystatAPI.BodystatAPI.ESRawData.record 112

BodystatAPI.BodystatAPI.ESRawData.ulFirstTestNum 112

BodystatAPI.BodystatAPI.ESReadStoredTestData 142

BodystatAPI.BodystatAPI.ESReadTestEquistat 143

BodystatAPI.BodystatAPI.ESReadTestEquistat@out int@out int@out int@out int@out int@out int@out int@out float 76

BodystatAPI.BodystatAPI.ESResults 112

BodystatAPI.BodystatAPI.ESResults structure 112

BodystatAPI.BodystatAPI.ImpData 114

BodystatAPI.BodystatAPI.ImpData structure 114

BodystatAPI.BSAnalysisMode 145

BodystatAPI.BSAnalysisMode enumeration 145

BodystatAPI.BSDeviceFamily 146

BodystatAPI.BSDeviceFamily enumeration 146

BodystatAPI.BSDeviceModel 147

BodystatAPI.BSDeviceModel enumeration 147

BodystatAPI.BSError 149

BodystatAPI.BSError enumeration 149

BodystatAPI.BSGender 151

BodystatAPI.BSGender enumeration 151

BodystatAPI.BSMeasurement structure 84
- about BodystatAPI.BSMeasurement structure 84
- fPA_50kHz 86
- fWeight 87
- fX_50kHz 87
- iActivity 87
- iAge 88
- iDeviceModel 88
- iFrequencies 88
- iGender 89
- iHeight 89
- iHip 89
- iR_50kHz 90
- iWaist 90
- iZ_100kHz 90
- iZ_200kHz 91
- iZ_50kHz 91
- iZ_5kHz 91
- llTestDate 92
- pMultifreqData 92
- tTestDate 93
- ulDeviceSerialNumber 92

BodystatAPI.BSRawData structure 96
- about BodystatAPI.BSRawData structure 96
- iRecordArraySize 97
- iTotalNumRecs 98
- New 97

- record 98
- ulFirstTestNum 99

BodystatAPI.ESMeasurement structure 101
- about BodystatAPI.ESMeasurement structure 101
- fPA_50kHz 103
- iBCScore 103
- iDeviceModel 104
- iHeight 104
- iZ_140kHz 104
- iZ_16kHz 105
- iZ_200kHz 105
- iZ_24kHz 105
- iZ_280kHz 106
- iZ_50kHz 106
- iZ_5kHz 106
- llTestDate 107
- tTestDate 107
- ulDeviceSerialNumber 107

BodystatAPI.ESRawData structure 109
- about BodystatAPI.ESRawData structure 109
- iRecordArraySize 111
- iTotalNumRecs 111
- New 110
- record 112
- ulFirstTestNum 112

BodystatCSWinForms 205

BodystatCSWinForms namespace 205
- Classes 205

BodystatCSWinForms.MainForm 205

BodystatCSWinForms.MainForm.MainForm 206

BodystatCSWinForms.MainForm.PopulateMeasurementList 207

BodystatCSWinForms.ResultsForm 209

BodystatCSWinForms.ResultsForm.InitialiseFromMeasurement 210

BodystatCSWinForms.ResultsForm.InitialiseSampleMeasurement 211

BodystatCSWinForms.ResultsForm.PopulateListCtrl 212

BodystatCSWinForms.ResultsForm.ResultsForm 209

BodystatCSWinForms.ResultsForm.UseBSMeasurement 215

BodystatSDK.cs 80

BodystatSDK.h 55

BODYSTATSDK_API 53

BODYSTATSDK_API macro 53

BodystatVBNetWinFormSample 217

BodystatVBNetWinFormSample namespace 217
- Classes 217

BodystatVBNetWinFormSample.FrmBodystatMain 217

BodystatVBNetWinFormSample.FrmBodystatMain.btnBtAuthenticateButton_Click 219

BodystatVBNetWinFormSample.FrmBodystatMain.btnBtAutoSetupButton_Click 220

BodystatVBNetWinFormSample.FrmBodystatMain.btnBtDeviceInfoButton_Click 220

BodystatVBNetWinFormSample.FrmBodystatMain.btnBtSearchButton_Click 221

BodystatVBNetWinFormSample.FrmBodystatMain.btnBtStackInfoButton_Click 222

BodystatVBNetWinFormSample.FrmBodystatMain.btnBtStatusButton_Click 222

BodystatVBNetWinFormSample.FrmBodystatMain.btnBtUnauthenticateButton_Click 223

BodystatVBNetWinFormSample.FrmBodystatMain.btnComConnectButton_Click 223

BodystatVBNetWinFormSample.FrmBodystatMain.btnComReadCalibrationTimeButton_Click 224

BodystatVBNetWinFormSample.FrmBodystatMain.btnComReadCurrentTimeButton_Click 225

BodystatVBNetWinFormSample.FrmBodystatMain.btnComReadModelVersionButton_Click 226

BodystatVBNetWinFormSample.FrmBodystatMain.btnComReadPrinterAddrButton_Click 227

BodystatVBNetWinFormSample.FrmBodystatMain.btnComReadProtocolInfoButton_Click 228

BodystatVBNetWinFormSample.FrmBodystatMain.btnComReadSerialNumberButton_Click 229

BodystatVBNetWinFormSample.FrmBodystatMain.btnComReadStoredTestDataButton_Click 230

BodystatVBNetWinFormSample.FrmBodystatMain.btnComReadTestMeasurementButton_Click 231

BodystatVBNetWinFormSample.FrmBodystatMain.btnComWriteCurrentTimeButton_Click 232

BodystatVBNetWinFormSample.FrmBodystatMain.btnResultsButton_Click 233

BodystatVBNetWinFormSample.FrmBodystatMain.FrmBodystatMain_Load 234

BodystatVBNetWinFormSample.FrmBodystatMain.iTimeout 219

BodystatVBNetWinFormSample.FrmBodystatMain.lstviewTestMeasurementList_DoubleClick 235

BodystatVBNetWinFormSample.FrmBodystatMain.PopulateMeasurementList 235

BodystatVBNetWinFormSample.FrmBodystatResults 237

BodystatVBNetWinFormSample.FrmBodystatResults.bsmMeasurement 238

BodystatVBNetWinFormSample.FrmBodystatResults.btnCalculateResultsButton_Click 239

BodystatVBNetWinFormSample.FrmBodystatResults.btnCopytoclipboardButton_Click 240

BodystatVBNetWinFormSample.FrmBodystatResults.bUseActualMeasurment 238

BodystatVBNetWinFormSample.FrmBodystatResults.dtTestDate 238

BodystatVBNetWinFormSample.FrmBodystatResults.FrmBodystatResults_Load 242

BodystatVBNetWinFormSample.FrmBodystatResults.InitialiseFromMeasurement 242

BodystatVBNetWinFormSample.FrmBodystatResults.InitialiseSampleMeasurement 243

BodystatVBNetWinFormSample.FrmBodystatResults.PopulateListCtrl 244

BodystatVBNetWinFormSample.FrmBodystatResults.UseBSMeasurement 247

BS1500_BT enumeration member 32, 147

BS1500_DIU enumeration member 32, 147

BS1500_Family enumeration member 30, 146

BS1500_OPTO enumeration member 32, 147

BSAnalysisModeBC enumeration member 30, 145

BSAnalysisModeBoth enumeration member 30, 145

BSAnalysisModeHN enumeration member 30, 145

BSAnalysisModeNone enumeration member 30, 145

BSDualScan_DIU enumeration member 32, 147

BSDualScan_Family enumeration member 30, 146

BSDualScan_OPTO enumeration member 32, 147

BSEquistatLite_BT enumeration member 32, 147

BSEquistatLite_Family enumeration member 30, 146

BSEquistatPro_BT enumeration member 32, 147

BSEquistatPro_Family enumeration member 30, 146

BSFemale enumeration member 35, 151

BSMale enumeration member 35, 151

BSMDD_BT enumeration member 32, 147

BSMDD_DIU enumeration member 32, 147

BSMDD_Family enumeration member 30, 146

BSMDD_OPTO enumeration member 32, 147

BSMultiScan_DIU enumeration member 32, 147

BSMultiScan_Family enumeration member 30, 146

BSMultiScanTouch enumeration member 32, 147

BSQuadScan_BT enumeration member 32, 147

BSQuadScan_DIU enumeration member 32, 147

BSQuadScan_Family enumeration member 30, 146

BSQuadScanTouch enumeration member 32, 147

BSUnknown enumeration member 32, 147

BSUnknown_Family enumeration member 30, 146

C

C# .NET Wrapper 56
- Files 79

CBodystatMFCDlg 156

CBodystatMFCDlg class 156
- about CBodystatMFCDlg class 156
- CBodystatMFCDlg 158
- DoDataExchange 165
- m_bsRawData 159
- m_btnAuthenticate 159
- m_btnAutoSetup 159
- m_btnConnect 160
- m_btnDisconnect 160
- m_btnReadCalibrationTime 160
- m_btnReadCurrentTime 161
- m_btnReadModelVersion 161

m_btnReadPrinterAddr 161
m_btnReadProtocolInfo 162
m_btnReadSerialNumber 162
m_btnReadStoredTestData 162
m_btnReadTestMeasurement 162
m_btnUnauthenticate 163
m_btnWriteCurrentTime 163
m_hIcon 163
m_iTimeout 164
m_strComPort 164
m_TestMeasurementListCtrl 164
OnClickedBtAuthenticateButton 165
OnClickedBtAutoSetupButton 166
OnClickedBtDeviceInfoButton 167
OnClickedBtSearchButton 167
OnClickedBtStackInfoButton 168
OnClickedBtStatusButton 169
OnClickedBtUnauthenticateButton 169
OnClickedComConnectButton 170
OnClickedComReadCalibrationTimeButton 170
OnClickedComReadCurrentTimeButton 171
OnClickedComReadModelVersionButton 172
OnClickedComReadPrinterAddrButton 173
OnClickedComReadProtocolInfoButton 174
OnClickedComReadSerialNumberButton 175
OnClickedComReadStoredTestDataButton 175
OnClickedComReadTestMeasurementButton 176
OnClickedComWriteCurrentTimeButton 177
OnClickedResultsButton 178
OnDblclkTestMeasurementList 179
OnInitDialog 179
OnPaint 180
OnQueryDragIcon 181
OnSysCommand 181
PopulateMeasurementList 182
ShowResultsDialog 184
CBodystatMFCDlg::CBodystatMFCDlg 158
CBodystatMFCDlg::DoDataExchange 165
CBodystatMFCDlg::m_bsRawData 159
CBodystatMFCDlg::m_btnAuthenticate 159
CBodystatMFCDlg::m_btnAutoSetup 159
CBodystatMFCDlg::m_btnConnect 160
CBodystatMFCDlg::m_btnDisconnect 160
CBodystatMFCDlg::m_btnReadCalibrationTime 160
CBodystatMFCDlg::m_btnReadCurrentTime 161
CBodystatMFCDlg::m_btnReadModelVersion 161
CBodystatMFCDlg::m_btnReadPrinterAddr 161
CBodystatMFCDlg::m_btnReadProtocolInfo 162
CBodystatMFCDlg::m_btnReadSerialNumber 162
CBodystatMFCDlg::m_btnReadStoredTestData 162
CBodystatMFCDlg::m_btnReadTestMeasurement 162
CBodystatMFCDlg::m_btnUnauthenticate 163
CBodystatMFCDlg::m_btnWriteCurrentTime 163
CBodystatMFCDlg::m_hIcon 163
CBodystatMFCDlg::m_iTimeout 164
CBodystatMFCDlg::m_strComPort 164
CBodystatMFCDlg::m_TestMeasurementListCtrl 164
CBodystatMFCDlg::OnClickedBtAuthenticateButton 165
CBodystatMFCDlg::OnClickedBtAutoSetupButton 166
CBodystatMFCDlg::OnClickedBtDeviceInfoButton 167
CBodystatMFCDlg::OnClickedBtSearchButton 167
CBodystatMFCDlg::OnClickedBtStackInfoButton 168
CBodystatMFCDlg::OnClickedBtStatusButton 169
CBodystatMFCDlg::OnClickedBtUnauthenticateButton 169
CBodystatMFCDlg::OnClickedComConnectButton 170
CBodystatMFCDlg::OnClickedComReadCalibrationTimeButton 170
CBodystatMFCDlg::OnClickedComReadCurrentTimeButton 171
CBodystatMFCDlg::OnClickedComReadModelVersionButton 172
CBodystatMFCDlg::OnClickedComReadPrinterAddrButton 173
CBodystatMFCDlg::OnClickedComReadProtocolInfoButton 174
CBodystatMFCDlg::OnClickedComReadSerialNumberButton 175
CBodystatMFCDlg::OnClickedComReadStoredTestDataButton 175
CBodystatMFCDlg::OnClickedComReadTestMeasurementButton 176
CBodystatMFCDlg::OnClickedComWriteCurrentTimeButton 177
CBodystatMFCDlg::OnClickedResultsButton 178
CBodystatMFCDlg::OnDblclkTestMeasurementList 179
CBodystatMFCDlg::OnInitDialog 179

CBodystatMFCDlg::OnPaint 180
CBodystatMFCDlg::OnQueryDragIcon 181
CBodystatMFCDlg::OnSysCommand 181
CBodystatMFCDlg::PopulateMeasurementList 182
CBodystatMFCDlg::ShowResultsDialog 184
CBodystatResultsDlg 185
CBodystatResultsDlg class 185
    ~CBodystatResultsDlg 186
    about CBodystatResultsDlg class 185
    CBodystatResultsDlg 186
    CopyToClipboard 194
    DoDataExchange 196
    GetSystemErrorMessage 197
    m_bModeBodyComposition 187
    m_bModeHydration 188
    m_dtTestDate 188
    m_fPhaseAngle50K 188
    m_fReactance50K 189
    m_fWeight 189
    m_iActivityLevel 189
    m_iAge 190
    m_iGender 190
    m_iHeight 190
    m_iHip 191
    m_iImpedance100K 191
    m_iImpedance200K 191
    m_iImpedance50K 192
    m_iImpedance5K 192
    m_iModel 192
    m_iResistance50K 193
    m_iWaist 193
    m_ModelComboCtrl 193
    m_ResultsListCtrl 194
    OnClickedCalculateResultsButton 198
    OnClickedCopyToClipboardButton 200
    OnInitDialog 200
    PopulateListCtrl 201
CBodystatResultsDlg::~CBodystatResultsDlg 186
CBodystatResultsDlg::CBodystatResultsDlg 186
CBodystatResultsDlg::CopyToClipboard 194
CBodystatResultsDlg::DoDataExchange 196
CBodystatResultsDlg::GetSystemErrorMessage 197
CBodystatResultsDlg::m_bModeBodyComposition 187
CBodystatResultsDlg::m_bModeHydration 188
CBodystatResultsDlg::m_dtTestDate 188
CBodystatResultsDlg::m_fPhaseAngle50K 188
CBodystatResultsDlg::m_fReactance50K 189
CBodystatResultsDlg::m_fWeight 189
CBodystatResultsDlg::m_iActivityLevel 189
CBodystatResultsDlg::m_iAge 190
CBodystatResultsDlg::m_iGender 190
CBodystatResultsDlg::m_iHeight 190
CBodystatResultsDlg::m_iHip 191
CBodystatResultsDlg::m_iImpedance100K 191
CBodystatResultsDlg::m_iImpedance200K 191
CBodystatResultsDlg::m_iImpedance50K 192
CBodystatResultsDlg::m_iImpedance5K 192
CBodystatResultsDlg::m_iModel 192
CBodystatResultsDlg::m_iResistance50K 193
CBodystatResultsDlg::m_iWaist 193
CBodystatResultsDlg::m_ModelComboCtrl 193
CBodystatResultsDlg::m_ResultsListCtrl 194
CBodystatResultsDlg::OnClickedCalculateResultsButton 198
CBodystatResultsDlg::OnClickedCopyToClipboardButton 200
CBodystatResultsDlg::OnInitDialog 200
CBodystatResultsDlg::PopulateListCtrl 201
ComErrorConfiguringPort enumeration member 33, 149
ComErrorConnecting enumeration member 33, 149
ComErrorOpeningPort enumeration member 33, 149
ComErrorProcessingCommand enumeration member 33, 149
ComErrorReading enumeration member 33, 149
ComErrorUnsupportedCommand enumeration member 33, 149
ComErrorUserAborted enumeration member 33, 149
ComErrorWriting enumeration member 33, 149

F

FAQs 152
FrmBodystatMain class 217
    about FrmBodystatMain class 217
    btnBtAuthenticateButton_Click 219
    btnBtAutoSetupButton_Click 220
    btnBtDeviceInfoButton_Click 220
    btnBtSearchButton_Click 221
    btnBtStackInfoButton_Click 222

btnBtStatusButton_Click 222
btnBtUnauthenticateButton_Click 223
btnComConnectButton_Click 223
btnComReadCalibrationTimeButton_Click 224
btnComReadCurrentTimeButton_Click 225
btnComReadModelVersionButton_Click 226
btnComReadPrinterAddrButton_Click 227
btnComReadProtocolInfoButton_Click 228
btnComReadSerialNumberButton_Click 229
btnComReadStoredTestDataButton_Click 230
btnComReadTestMeasurementButton_Click 231
btnComWriteCurrentTimeButton_Click 232
btnResultsButton_Click 233
FrmBodystatMain_Load 234
iTimeout 219
lstviewTestMeasurementList_DoubleClick 235
PopulateMeasurementList 235

FrmBodystatResults class 237
about FrmBodystatResults class 237
bsmMeasurement 238
btnCalculateResultsButton_Click 239
btnCopytoclipboardButton_Click 240
bUseActualMeasurment 238
dtTestDate 238
FrmBodystatResults_Load 242
InitialiseFromMeasurement 242
InitialiseSampleMeasurement 243
PopulateListCtrl 244
UseBSMeasurement 247

I

ImpData structure 49
about ImpData structure 49
fFrequency 50
fImpedance 50
fPhaseAngle 51
ImZ 51
ReZ 51

InvalidActivity enumeration member 33, 149
InvalidAge enumeration member 33, 149
InvalidDeviceFamily enumeration member 33, 149
InvalidDeviceMode enumeration member 33, 149
InvalidDeviceModel enumeration member 33, 149
InvalidGender enumeration member 33, 149
InvalidHeight enumeration member 33, 149
InvalidHip enumeration member 33, 149
InvalidImpedanceZ100 enumeration member 33, 149
InvalidImpedanceZ200 enumeration member 33, 149
InvalidImpedanceZ5 enumeration member 33, 149
InvalidImpedanceZ50 enumeration member 33, 149
InvalidParamSupplied enumeration member 33, 149
InvalidPhaseAnglePA50 enumeration member 33, 149
InvalidReactanceX50 enumeration member 33, 149
InvalidResistanceR50 enumeration member 33, 149
InvalidWaist enumeration member 33, 149
InvalidWeight enumeration member 33, 149

M

M_PI 54
M_PI macro 54
MainForm class 205
about MainForm class 205
MainForm 206
PopulateMeasurementList 207
MainForm.cs 216

N

NoError enumeration member 33, 149

R

Redistribution 154
ResultsForm class 209
about ResultsForm class 209
InitialiseFromMeasurement 210
InitialiseSampleMeasurement 211
PopulateListCtrl 212
ResultsForm 209
UseBSMeasurement 215
ResultsForm.cs 216

S

Samples 155

V

Visual Basic .NET wrapper 80
  Classes 81
  Structs, Records, Enums 145