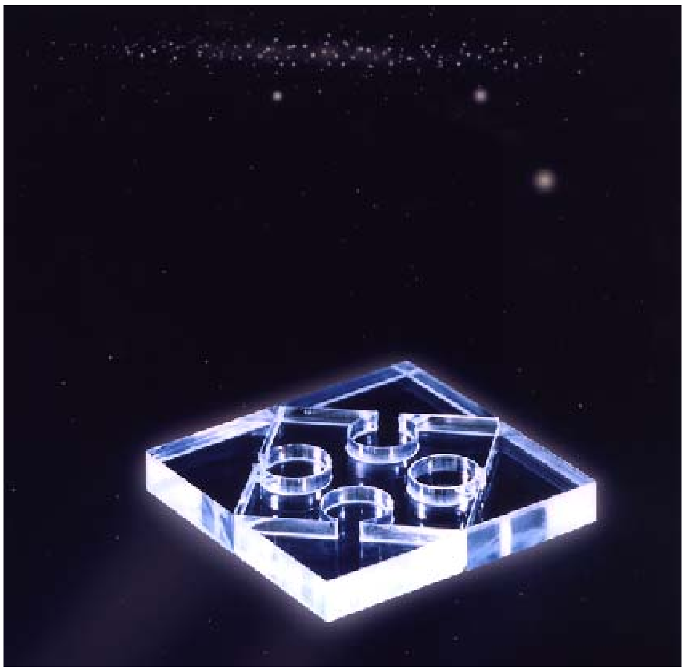
Axyz

OLE-Commands for Script Programming

CDM, STM/MTM, LTM, DB Access



Leica Geosystems AG, Unterentfelden

Proprietary information

The ***Axyz*** program has been developed by Leica Geosystems AG Unterentfelden Switzerland and Leica Geosystems Inc. Atlanta USA.

This manual was produced using Microsoft Word 97, but saved as Word 6.0 Document.

Document reference data

Title:	***Axyz*** OLE-Commands for Script Programming
Reference code:	575 701
Software version:	Release 1.4.2
Document release date:	January 2005

Contents

1. Axyz OLE - Automation Commands

1.1. What’s changed since Version 1.4

1.1.1. CDM

New: No changes

Functions, which are not documented but appear in the interface, should not be used. They are kept in the interface for compatibility or internal testing reason.

- ShowAxyzMain
- ReadPoints
- GetNextPoint
- ReadShapes
- GetNextShape
- ReadMeasurements
- GetNextMeasurement
- ReadStation
- GetNextStation

1.1.2. STM/MTM

New:

- MTM/STM Registered Windows Messages

0 SYSMSG_ATR_ERROR_CONDITION “ATRErrorCondition"
1 SYSMSG_ATR_STATUS_CHANGE "ATRStatusChange"

- ClearATRErrorCondition
- EnableATRErrorMessageBox

1.1.3. LTM

New: No changes

Functions, which are not documented but appear in the interface, should not be used. They are kept in the interface for compatibility or internal testing reason:

- ShowTestAlignDlg
- ShowAboutDlg
- SelectMode
- ShowEstablishDistADM
- ChangeViewFull

- RefreshViews
- RefreshStatusView
- ShowGoBirdBath

1.1.4.Database

New: No Changes

1.2. General Programming Concepts

This document describes a number of commands to extend any programming language capable of using OLE Automation to control a large number of functions of the ***Axyz*** Core Module program and the Theodolite Manager STM/MTM.

If you should notice that Axyz offers some other functionality for which you don't find an appropriate OLE function, please contact us. We will do our best to offer more functionality as requests come in for.

In case you are using Visual Basic, we recommend to use Version 6.0 (Professional or Enterprise edition, 32bit set-up). With previous versions the passing of array parameters might be critical.

The "Integer" type of a parameter is the 2 Byte signed VB "Integer" data type. VC++ scripts use the 2 Byte signed "short" type.

1.3. Initialization

In order to use any of the OLE Automation command in a Visual Basic program, a short initialization part must be called. This initialization is for all the scripts identical, and can therefore be kept in a separate file with the extension BAS, and included in all subsequent script files. It contains a hand full of declarations and several procedures.

For most of your projects you can use the "Show Form on top.Bas" file, which is located in the *Scripts* subdirectory of the ***Axyz*** data directory.

A call to one of the system DLL's to make the basic script always appear on top may look as follows:

```
Private Declare Function SetWindowPos Lib "user32" (ByVal hwnd As
Long, ByVal hWndInsertAfter As Long, ByVal x As Long, ByVal y As Long,
ByVal cx As Long, ByVal cy As Long, ByVal wFlags As Long) As Long

Private Sub Form_Load()
'Put this window on top of all
Status = SetWindowPos(hwnd, -1, 0, 0, 0, 0, 3)
....
....
....
```

Establishing the connections to the ***Axyz*** CoreModule CDM and Theodolite Manager STM/MTM may look like the following source code:

```
Public Sub ConnectAxyz(ByVal TheoTask As Boolean, ByVal_
TrackerTask As Boolean, ByVal ActiveLanguage As String)
' Sub to link Axyz with current script
' IN:   TheoTask = Flag for theomanager to be initialized
'       TrackerTask = Flag for trackermanager to be
'       initialized
'       ActiveLanguage = Language setting according to
'       current selection
'
' Declare some local variables
Dim Counter As Integer
Dim BoxTitle As String
Dim ConnectionTask(2), TaskID(2) As String
Dim TaskRequired(2) As Boolean

' Initialize variables
ConnectionTask(1) = "theoman.exe"
TaskID(1) = "STM/MTM"
TaskRequired(1) = TheoTask
ConnectionTask(2) = "ltm.exe"
TaskID(2) = "LTM"
TaskRequired(2) = TrackerTask

' Connect CoreModule
Set CoreM = CreateObject("Axyz.Document")

' Connect task and quit script if prompted No
For Counter = 1 To 2
  If CoreM.IsTaskRunning(ConnectionTask(Counter)) = False_
  And (TaskRequired(Counter) = True) Then
    If InStr(1, ActiveLanguage, "Deutsch", 1) Then
        Msg = "Makro beendet ! Starten Sie bitte zuerst_
        das Modul " + TaskID(Counter) + " und danach_
        dieses Script !"
        BoxTitle = "Verbindung nach Axyz"
    ElseIf InStr(1, ActiveLanguage, "Français", 1) Then
        Msg = "Macro terminée ! Au premiér démarrez le_
        module " + TaskID(Counter) + " et après démarrez_
        la Macro !"
        BoxTitle = "Connecter Axyz"
    Else
        Msg = "Script aborted ! First re-start the module_
         " + TaskID(Counter) + ", then re-start this_
        Script !"
        BoxTitle = "Link with Axyz"
    End If
    Style = vbCritical + vbOKOnly
    Response = MsgBox(Msg, Style, BoxTitle)
    End
  End If
Next
```

```
If TheoTask = True Then
    Set TheoM = CreateObject("Theoma.Document")
    TheoM.SetPointID "ALL", "Wait", "Dummy PointID"
ElseIf TrackerTask = True Then
    Set TrackerM = CreateObject("ltm.Document")
End If

End
```

To run an ***Axyz*** Script, the ***Axyz*** CoreModule CDM and Theodolite Manager STM/MTM must be already activated. After each High Level Command ***Axyz*** gets the focus. In order to bring script messages to the front after such a High Level Command we recommend to use the command *Me*.SetFocus.

1.4. Calling Conventions

1.4.1. General syntax

There are two quite distinct ways to call the commands:

- As a function, returning a Value:
  Value = AnyCommand (Param1, Param2)

in this case, the parameters must be separated by commas and put between brackets.

- As a call:
  AnyCommand Param1, Param2

in this case, the parameters are separated by commas but **not** put between brackets.

1.4.2. STM/MTM specific syntax

All commands, that allow an action to be carried out on more than one sensor at any time, have a string to select the stations. The syntax of this string is :

“all”	:	all stations will be called
“1 2”	:	only the selected stations will be called

In any case, only the stations that can handle the command will accept it.
Examples:

- A call to **OpenTheoView** can only be accepted from a station that is oriented.
- A call to **ShowLevelTheodolite** can only be accepted from a station that has no previous measurements.
- A call for **SetTryMode** applies only if a view is open and the view has not in continuous mode active.
-

1.4.3. LTM Specific syntax

There is nothing unique about the syntax for LTM - commands beyond what is described in section 1.4.1 above.

1.4.4. DB Access Specific syntax

There is nothing unique about the syntax for database access commands beyond what is described in section 1.4.1 above.

1.5. Using a Message Blaster (MB) with some LTM and MTM/STM commands

The commands StartMeasure, GoHome and GoLOcationPoint are giving back the control to VB immediately after starting it. As soon as the tracker operation has been finished, the function sends out a message. The Message Blaster can capture this message in your script and e.g. a next measurement can be started. The time between sending the command and getting back the message can be used for other activities. But no LTM OLE command may be started during this period.

The MTM/STM module (starting with version 1.4.1) sends information concerning the status and possible error conditions of the ATR functionality using this method.

1.5.1. The principle of how to use the MB32 OLE Control

The MessageBlaster32 was designed to handle Windows Messages in VB program easily. You can use it in 32bit VB 6.0. Once the MessageBlaster32 OCX is inserted, you can watch for designated Window Messages. When a specified message arrives, the MB32 will fire "Message" event, which can be easily handled in the VB program.
This OCX is free for any purpose, and anybody can distribute it freely.
Enjoy!
Naohiko Hashi, May 12, 1997

The self ectracting MessageBlaster file (Mbocx32b.exe) is stored on the Axyz CD in the special directory ***MsgBlast***. In addition to our example in 1.5.2 you can find here another example of a demo script. The file to be inserted is *msgblst32.ocx* . For registration of the ocx please read the file msgblst.txt carefully.

1.5.2. LTM Registered Windows Messages

The example below shows how to use the Message Blaster in conjuction with the commands **StartMeasure, GoHome and GoLocationPoint**.

VB Script (Code of Form):

```
Dim LTM As Object

'User message, used for internal message handling
Const WM_USER = &H400

' const declaration for "internal" commands, add new commands as
needed
Const INTERNAL_CMD_MEASURE = 1

'Message values from LTM
Const LTOLE_NONE = 0
Const LTOLE_GOLOCATIONPOINT = 1
Const LTOLE_GOLOCATIONPOINTADM = 2
Const LTOLE_GOHOME = 3
Const LTOLE_STARTMEASURE = 4

'Return values from LTM
Const LTOLE_OK = 0
Const LTOLE_NOK = 1
Const LTOLE_TIMEOUT = 2

Private Declare Function SetWindowPos Lib "user32" (ByVal hwnd As
Long, ByVal hWndInsertAfter As Long, ByVal x As Long, ByVal y As Long,
ByVal cx As Long, ByVal cy As Long, ByVal wFlags As Long) As Long

Private Declare Function PostMessage Lib "user32" Alias "PostMessageA"
(ByVal hwnd As Long, ByVal wMsg As Long, ByVal wParam As Long, ByVal
lParam As Long) As Long

Private Declare Function RegisterWindowMessage Lib "user32" Alias
"RegisterWindowMessageA" (ByVal Name As String) As Long

Private Sub Form_Load()
'Put this window on top of all
Status = SetWindowPos(hwnd, -1, 0, 0, 0, 0, 3)

' initialize the Message Blaster OCX to receive external messages
MsgBlst1.MsgList(0) =
RegisterWindowMessage("LTMOleAutomationStatusMessage")
MsgBlst1.MsgPassage(0) = 0
MsgBlst1.hWndTarget = hwnd

' initialize the Message Blaster OCX to receive internal messages
MsgBlst1.MsgList(1) = WM_USER
MsgBlst1.MsgPassage(1) = 0

'Create an instance of LTM
Set LTM = CreateObject("ltm.document")

'do not show any error messages
LTM.ShowErrorMessages False

'Put initial values in coordinate fields
XCoord.Text = 1000#
YCoord.Text = 1000#
ZCoord.Text = 500#

End Sub
```

```
'Message handler for LTM messages
Private Sub MsgBlst1_Message(ByVal MsgVal As Long, ByVal wParam As
Long, ByVal lParam As Long, lplRetVal As Long)

    'handler for the "internal" messages, use this to dispatch
    'commands using "PostMessage" calls.
    If (MsgVal = WM_USER) Then
        If (lParam = INTERNAL_CMD_MEASURE) Then
            StartMeas_Click
        End If

    'handler for "Registered Window Messages" as generated by LTM
    ElseIf (MsgVal =
        RegisterWindowMessage("LTMOleAutomationStatusMessage")) Then

        'Program flow gets here, if a GoLocation command has finished
         If ((lParam = LTOLE_GOLOCATIONPOINT) Or (lParam =
              LTOLE_GOLOCATIONPOINTADM)) Then
            If (wParam = LTOLE_OK) Then
                StatusText.Text = "Positioning is finished"
                ' call the Measure command (post internal message!)
                PostMessage hwnd, WM_USER, 0, INTERNAL_CMD_MEASURE
            Else
                StatusText.Text = "Positioning Not OK"
            End If

        'Program flow gets here, if a GoHome command has finished
        ElseIf (lParam = LTOLE_GOHOME) Then
            StatusText.Text = "Goto BirdBath is finished"

        'Program flow gets here, if StartMeasure command has finished
        ElseIf (lParam = LTOLE_STARTMEASURE) Then
            StatusText.Text = "Measurement is finished"

        End If
    End If
End Sub

Private Sub Exit_Click()

    End
End Sub

Private Sub GoHome_Click()

    StatusText.Text = "Goto BirdBath started, please wait..."
    LTM.GoHome 1
End Sub

Private Sub GoLocation_Click()

    StatusText.Text = "GoLocation started, please wait..."
    LTM.GoLocationPoint 1, XCoord.Text, YCoord.Text, ZCoord.Text,
         False
End Sub

Private Sub StartMeas_Click()

    StatusText.Text = "Measurement started, please wait..."
    LTM.StartMeasure
End Sub
```

1.5.3.MTM/STM Registered Windows Messages

MTM/STM (starting with version 1.4.1) uses the following two Registered Windows messages:

SYSMSG_ATR_ERROR_CONDITION “ATRErrorCondition"

This message is sent, when an ATR related error occurs. The error number and error message text is coming directly from the theodolite. (See the theodolite user manual for all possible error numbers).
A typical error received, if the sensor can not find a reflector, is error 565. (Message from sensor: "565/No prism found or bad conditions. Aim accurately at the prism and retry.")

SYSMSG_ATR_STATUS_CHANGE "ATRStatusChange"

This message is generated when the ATR status changes. The possible values are:

0	:	No ATR Available
1	:	ATR Disabled
2	:	ATR Enabled
3	:	ATR Lock Enabled
4	:	ATR Locked

2. CDM - High Level Commands

2.1. General Remarks

This section describes the commands, that call the built-in dialog boxes of the Core Module. Most of them take (optional) arguments to supply with initial default values.
Note that it is not possible to omit default parameters for Visual Basic functions. Hence, empty strings or zeros (case of numerical values) must be passed. See the examples below.
Most parameters of the 'High Level commands' are optional. They just offer a way to initialize the dialog box. However, since Visual Basic does not support optional parameters in terms of omitting them, at least empty/null values (explicit defaults) must be passed for all parameters.
These 'NULL' values are as follows:

• Strings:	"" (empty string)
• Integers, Longs:	0
• Boolean:	False or True
• Doubles:	0.0

Rule of thumb: The parameters of the 'High Level Commands' generally match exactly the GUI controls of the related dialog box. Thus the meaning of these parameters are not explained in deep detail within this section. If in doubt, call the appropriate dialog box and see its 'Help'.

2.2. General Commands

2.2.1.ShowAxyzMain

obsolete, see ShowAxyzCore

2.2.2.ShowAxyzCore

This function replaces the obsolete one "ShowAxyzMain". With TRUE parameter, the CDM is shown, with FALSE it is minimized.

Input Parameter	
Status	(Boolean)
Return Value	
bStatus	(Boolean)
Example	bStatus = CoreM.**ShowAxyzCore**(True)

2.2.3. ShowFileDelete

This function opens the “File-Delete” dialog.

Return Value

bStatus	(Boolean)
Example	bStatus = CoreM.**ShowFileDelete**()

2.3. Settings Commands

2.3.1. ShowSetWorkpiece

Call the “Current Workpiece” dialog box . The script language program flow will pause, until the dialog box is closed by the user.

Return Value

Status	(Boolean)
Example	bStatus = CoreM.**ShowSetWorkpiece**()

2.3.2. ShowSetUnits

Call the “Units” dialog box . The script language program flow will pause, until the dialog box is closed by the user.

Return Value

Status	(Boolean)
Example	CoreM.**ShowSetUnits**

2.3.3. ShowSetWarnings

Call the “Analysis Warnings” dialog box. The script language program flow will pause, until the dialog box is closed by the user.

Return Value

Status	(Boolean)
Example	bStatus = CoreM.**ShowSetWarnings**()

2.3.4. ShowSetOnlineOutput

Call the “Online Output” dialog box. The script language program flow will pause, until the dialog box is closed by the user.

Return Value

Status	(Boolean)
Example	CoreM.**ShowSetOnlineOutput**

2.3.5. ShowSetConfirms

Call the “Confirmations” dialog box. The script language program flow will pause, until the dialog box is closed by the user.

<u>*Return Value*</u>	
Status	(Boolean)
<u>*Example*</u>	CoreM.**ShowSetConfirms**

2.3.6. ShowSetLabelAxis

Call the “Axis Labels” dialog box. The script language program flow will pause, until the dialog box is closed by the user.

<u>*Return Value*</u>	
Status	(Boolean)
<u>*Example*</u>	CoreM.**ShowSetLabelAxis**

2.3.7. ShowSetGeneral

Call the “General” dialog box. The script language program flow will pause, until the dialog box is closed by the user.

<u>*Return Value*</u>	
Status	(Boolean)
<u>*Example*</u>	CoreM.**ShowSetGeneral**

2.4. Tools Menu Commands

2.4.1. ShowMovePoints

Call the “Swap Points” dialog box. The script language program flow will pause, until the dialog box is closed by the user. All parameters are defaults only.

<u>*Input Parameter*</u>

Single point ID, SQL point ID statement or input filename	(String)
Reference Workpiece (if ordinary points to reference points)	(String)
Workpiece ID	(String)
Reference Coordinate System	(String)

<u>*Return Value*</u>

Status	(Boolean)
Example	CoreM.**ShowMovePoints** “”, “”, “”, “Default/BASE”

2.5. Coordinate System Commands

2.5.1. ShowActiveCoordSys

Call the “Active Coordinate System” dialog box. The script language program flow will pause, until the dialog box is closed by the user.

Return Value

Status	(Boolean)
Example	bOK = CoreM.**ShowActiveCoordSys**()

2.5.2. ShowCoordSystemType

Call the “Coordinate System Type” dialog box The script language program flow will pause, until the dialog box is closed by the user.

Return Value

Status	(Boolean)
Example	CoreM.**ShowCoordSystemType**

2.5.3. ShowScaleCoordSys

Call the “Scale” dialog box. The script language program flow will pause, until the dialog box is closed by the user.

Input Parameter

Starting Coordinate System ID	(String)
New Coordinate System ID	(String)
Scale Factor	(Double)
First auxiliary Point ID	(String)
Second auxiliary Point ID	(String)
Distance between auxiliary Points	(Double)
Comment	(String)

All parameters are optional. However, at least empty strings / zeros must be passed. See “General Remarks” at the beginning of this section.
Note that the current Coordinate System is always filled in as first parameter by default. It can be overridden by passing an ID different from an empty one.

Return Value

Status	(Boolean)

Example

bStatus = CoreM.**ShowScaleCoordSys**(“default/sc2“, ““, 2.0, ““, ““, 0, ““)
CoreM.**ShowScaleCoordSys** “default/xy“, “wp2/scx“, 0.0, “default/pt1“, “default/pt2“, 2.745, ““

2.5.4. ShowRotateCoordSys

Call the “Rotation” dialog box. The script language program flow will pause, until the dialog box is closed by the user.

Input Parameter

Starting Coordinate System ID	(String)
New Coordinate System ID	(String)
Rotation Angle	(Double)
Rotation Axis	(Integer)
Point to rotate about ID	(String)
Comment	(String)

The ‘Rotation Axis’ parameter can take the following values:
1 = X
2 = Y
3 = Z

All parameters are optional. However, at least empty strings / zeros must be passed. See “General Remarks” at the beginning of this section
Note that the current Coordinate System is always filled in as first parameter by default. It can be overridden by passing an ID different from an empty one.

Return Value

Status	(Boolean)

Example

bStatus = CoreM.**ShowScaleCoordSys** (““, ““,0, ““, ““, 0, ““)
CoreM.**ShowScaleCoordSys** “default/sc2“, “wp2/scx“, 0.0,”default/pt1“, “default/pt2“, 2.745, ““

2.5.5. ShowTranslateCoordSys

Call the “Translation” dialog box. The script language program flow will pause, until the dialog box is closed by the user.

<u>*Input Parameter*</u>

Starting Coordinate System ID	(String)
New Coordinate System ID	(String)
Translation along X	(Double)
Translation along Y	(Double)
Translation along Z	(Double)
Point ID to translate to	(String)
Comment	(String)

All parameters are optional. However, at least empty strings / zeros must be passed. See chapter 0 **Error! Unknown switch argument.** at the beginning of this section.
Note that the current Coordinate System is always filled in as first parameter by default. It can be overridden by passing an ID different from an empty one.

<u>*Return Value*</u>

Status	(Boolean)

<u>*Example*</u>

CoreM.**ShowTranslateCoordSys** “wp1/cs12“, ““,1,0,0,““, “this is a comment“

2.5.6. ShowAxisAlignCoordSys

Call the “Axis Alignment” dialog box . The script language program flow will pause, until the dialog box is closed by the user.

<u>*Input Parameter*</u>

New Coordinate System ID	(String)
Align Point 1	(String)
Control Point Coord X	(Double)
Control Point Coord Y	(Double)
Control Point Coord Z	(Double)
Align Point 2	(String)
Align Type2	(Integer)
Align Point 3	(String)
Align Type3	(Integer)
Horizontal Plane Flag	(Boolean)
Comment	(String)

The ‘Align Type’ parameters can take the following values:
1 = X
2 = Y
3 = Z
-1 = -X
-2 = -Y
-3 = -Z

All parameters are optional. However, at least empty strings / zeros must be passed. See chapter 0 **Error! Unknown switch argument.** at the beginning of this section.
Note that the current Coordinate System is always filled in as first parameter by default. It can be overridden by passing an ID different from an empty one.

Return Value

Status	(Boolean)

Example

CoreM.**ShowAxisAlignCoordSys** “wp1/ax1“, “wp1/pt1“, 0., 0. 0., “wp1/pt2“, 3, “wp1/pt3“, -2, False, “This is a comment“

2.5.7. ShowTransformCoordSys

Call the “Transformation” dialog box . The script language program flow will pause, until the dialog box is closed by the user.

Input Parameter

New Coordinate System ID	(String)
Update Flag	(Boolean)
Calculate Scale Flag	(Boolean)
Input Points	(String)
Mean Error Flag	(Boolean)
Prefix to remove	(String)
Prefix to add	(String)
Add to Point Number	(Long)

Return Value

Status	(Boolean)

All parameters are optional. However, at least empty strings / zeros must be passed. See chapter 0 **Error! Unknown switch argument.** at the beginning of this section.
The ‘Input Points’ parameter relates either to a filename or to a wildcard SQL statement. This file - if not in the current directory - must be specified by a full path. It is simply a text file, each line taking exactly one point id of the following syntax ‘refID/workpiece/pointID’.
If the given file is not found and if the passed string looks to be wildcard SQL statement, it is interpreted that way.

Example
CoreM.**ShowTransformCoordSys** “base/ct1“, False, False, “refWing”,
    True, ““, ““, 0

2.6. Analysis Commands

2.6.1. ShowAnalyzeParallel

Call the “Parallel…” dialog box. The script language program flow will pause, until the dialog box is closed by the user.

Input Parameter

Calculation Type	(Integer)
Element 1	(String)
Element 2	(String)
New Shape ID	(String)

The ‘Calculation Type’ parameter can take the following values:
0 = ParallelLine
1 = ParallelPlane

All parameters are optional. However, at least empty strings / zeros must be passed. See chapter 0 **Error! Unknown switch argument.** at the beginning of this section.

Return Value

Status	(Boolean)

Example
bStatus = CoreM.**ShowAnalyzeParallel**(1, “wp1/shp1“, “wp1/shp2“, ““)
CoreM.**ShowAnalyzeParallel** 0, ““, ““, ““

2.6.2. ShowAnalyzePerpend

Call the “Perpendicular” dialogue box. The script language program flow will pause, until the dialogue box is closed by the user.

Input Parameter

Calculation Type	(Integer)
Element 1	(String)
Element 2	(String)
New Shape ID	(String)

The ‘Calculation Type’ parameter can take the following values:
0 = PerpendPtShAxis
1 = PerpendPtShSurf
2 = PerpendShAxisShAxis

All parameters are optional. However, at least empty strings / zeros must be passed. See chapter 0 **Error! Unknown switch argument.** at the beginning of this section.

Return Value

Status	(Boolean)
Example	CoreM.**ShowAnalyzePerpend** 0, ““, ““, ““

2.6.3. ShowAnalyzeBisect

Call the “Bisect” dialog box. The script language program flow will pause, until the dialog box is closed by the user.

Input Parameter

Calculation Type	(Integer)
Element 1	(String)
Element 2	(String)
New Shape ID	(String)

The ‘Calculation Type’ parameter can take the following values:
Calculation Type:
0 = BisectPtPt
1 = BisectPtShAxis
2 = BisectPtPlane
3 = BisectShAxisShAxis
4 = BisectShAxisPlane
5 = BisectPlanePlane

All parameters are optional. However, at least empty strings / zeros must be passed. See chapter 0 **Error! Unknown switch argument.** at the beginning of this section.

Return Value

Status	(Boolean)
Example	CoreM.**ShowAnalyzeBisect** 0, "", "", ""

2.6.4. ShowAnalyzeProjection

Call the "Projection" dialog box. The script language program flow will pause, until the dialog box is closed by the user.

Input Parameter

Point to project	(String)
Shape	(String)
New Point ID	(String)

All parameters are optional. However, at least empty strings / zeros must be passed. See chapter 0 **Error! Unknown switch argument.** at the beginning of this section.

Return Value

Status	(Boolean)
Example	CoreM.**ShowAnalyzeProjection** "", "", ""

2.6.5. ShowAnalyzeIntersect

Call the "Intersect" dialog box . The script language program flow will pause, until the dialog box is closed by the user.

Input Parameter

Calculation Type	(Integer)
Element 1	(String)
Element 2	(String)
New Shape ID	(String)

The 'Calculation Type' parameter can take the following values:
Calculation Type:
0 = InterShAxisShAxis
1 = InterShAxisShSurf
2 = InterShSurfShSurf

All parameters are optional. However, at least empty strings / zeros must be passed. See chapter 0 **Error! Unknown switch argument.** at the beginning of this section.

Return Value

Status	(Boolean)

Example

CoreM.**ShowAnalyzeIntersect** 2, “wp1/shpere1“, “wp1/line1“, “wp2/pt1“

2.6.6. ShowAnalyzeDivideLine

Call the “Divide Line” dialog box . The script language program flow will pause, until the dialog box is closed by the user.

Input Parameter

Element 1	(String)
Element 2	(String)
Number of Division points	(Integer)
Starting ID of created points	(String)
Offset distance from first point	(Double)

All parameters are optional. However, at least empty strings / zeros must be passed. See chapter 0 **Error! Unknown switch argument.** at the beginning of this section.

Return Value

Status	(Boolean)

Example

CoreM.**ShowAnalyzeDivideLine** “wp1/pt1“, “wp1/pt2“, 1, “dp“, 0.

2.6.7. ShowAnalyzeCompareSets

Call the “Compare Sets” dialog box. The script language program flow will pause, until the dialog box is closed by the user.

Input Parameter

Input Points	(String)
Reference Coordinate system ID	(String)
Prefix to remove	(String)
Prefix to add	(String)
Point Number to add	(long)
‘Show out of tolerance only’ flag	(Boolean)
‘Tolerances from reference’ flag	(Boolean)

All parameters are optional. However, at least empty strings / zeros must be passed. See chapter 0 **Error! Unknown switch argument.** at the beginning of this section.
The 'Input Points' parameter relates either to a filename or to a wildcard SQL statement. This file - if not in the current directory - must be specified by a full path. It is simply a text file, each line taking exactly one point id of the following syntax 'refID/workpiece/pointID' (if the points are taken from the reference section) or 'workpiece/pointID[:<devNr>]' if taken from the normal section. Note that these two notations cannot be mixed within one file!
Note: 'devNr' only must be specified if > 0. Default is "no device number". Do not specify '0' for 'no device number'. Omit the 'devNr' instead.
If the given file is not found and if the passed string looks to be wildcard SQL statement, it is interpreted that way.

Return Value

Status	(Boolean)

Example

CoreM.**ShowAnalyzeCompareSets** "C:\axyz\script\cmpoints.dat",
"Default/BASE", "ref1/wp1", "base", 0, False, False
bOK = CoreM.**ShowAnalyzeCompareSet**("ref1/wp1/A*",
"Default/BASE", "ref1/wp1/", "simdat/", 0, True, False)
i.e. all points starting with *A* of the reference *ref1/wp1* to be compared with the matching points of the workpiece *simdat* in relation to the CS Default/BASE.

2.6.8. ShowAnalyzeTwoPoint

Call the "Two Point Analysis" dialog box. The script language program flow will pause, until the dialog box is closed by the user.

Input Parameter

Element 1	(String)
Element 2	(String)

All parameters are optional. However, at least empty strings / zeros must be passed. See chapter 0 **Error! Unknown switch argument.** at the beginning of this section.

Return Value

Status	(Boolean)
Example	CoreM.**ShowAnalyzeTwoPoint** "", ""

2.6.9. ShowAnalyzeHiddenPoint

Call the "Hidden Point" dialog box . The script language program flow will pause, until the dialog box is closed by the user.

Input Parameter

Hidden Point to calculate	(String)
Comment	(String)
'Auto Save' flag	(Boolean)

All parameters are optional. However, at least empty strings / zeros must be passed. See chapter 0 **Error! Unknown switch argument.** at the beginning of this section.
If the 'Auto Save' parameter is set, then not only a calculation is triggered on dialog pop-up - in addition the result will immediately be saved. (This is or was required for the Theodolite manager which makes use of this function through a OLE call.

Return Value

Status	(Boolean)

Example

bOK = CoreM.**ShowAnalyzeHiddenPoint**("wp1/HP1", "this is a hidden point", False)

2.6.10. ShowAnalyzeAngle

Call the "Angle" Dialog box. The script language program flow will pause, until the dialog box is closed by the user.

Input Parameter

Input Type1	(Integer)
Element 11	(String)
Element 12	(String)
Input Type2	(Integer)
Element 21	(String)
Element 22	(String)

All parameters are optional. However, at least empty strings / zeros must be passed. See chapter 0 **Error! Unknown switch argument.** at the beginning of this section.

The ‘Input Type’ parameters can take the following values:
0 = AngTwoPoint
1 = AngShapeAxis
2 = AngVector
Depending on the types, Element12/Element22 become obsolete (if element 11/21 is either a vector or a shape axis, i.e. not a point)

Return Value

Status	(Boolean)
Example	CoreM.**ShowAnalyzeAngle** 0, ““, ““0, ““, ““

bOK = **ShowAnalyzeAngle**(0, “wp1/pt1“, “wp1/pt1“0, “wp2/pt1“, “wp2/pt1“)

2.6.11. ShowAnalyzeDistance

Call the “Distance” Dialog box . The script language program flow will pause, until the dialog box is closed by the user.

Input Parameter

Calculation Type	(Integer)
Element 1	(String)
Element 2	(String)

All parameters are optional. However, at least empty strings / zeros must be passed. See chapter 0 **Error! Unknown switch argument.** at the beginning of this section.

The ‘Calculation Type’ parameter can take the following values:
0 = DistPtPt
1 = DistPtShAxis
2 = DistPtShSurf
3 = DistShAxisShAxis
4 = DistPlanePlane
5 = DistShAxisPlane

Return Value

Status	(Boolean)

Example

CoreM.**ShowAnalyzeDistance** 0, ““, ““

2.6.12. ShowAnalyzeArcDistance

Call the “Arc Distance” Dialog box. The script language program flow will pause, until the dialog box is closed by the user.

<u>*Input Parameter*</u>

Element 1	(String)
Element 2	(String)
Radius of Arc	(Double)

All parameters are optional. However, at least empty strings / zeros must be passed. See chapter 0 **Error! Unknown switch argument.** at the beginning of this section.

<u>*Return Value*</u>

Status	(Boolean)

<u>*Example*</u>

CoreM.**ShowAnalyzeArcDistance** “wp1/pt1“, “wp2/pt2“, 120.0

2.6.13. ShowAnalyzeShapefit

Call the “Shapefit” dialog box. The script language program flow will pause, until the dialog box is closed by the user.

<u>*Input Parameter*</u>

Shape type	(Integer)
Shape ID	(String)
Update Flag	(Boolean)
Thickness Correct Flag	(Boolean)
Input Points	(String)
Mean Error Flag	(Boolean)

All parameters - **except 'Shape type'** - are optional. However, at least empty strings / zeros must be passed. See "General Remarks" at the beginning of this section.
The 'Input Points' parameter relates either to a filename or to a wildcard SQL statement.
If a SQL statement, it has typically a syntax like 'workpiece1/plane*' (or 'workpiece1/plane*/*' in order to include also non-zero device points - Note that 'workpiece1/plane*' is equivalent to 'workpiece1/plane*/0'). This means to select all points whose name starts with 'plane' (plane1, plane2, plane3…) within the workpiece1.
A statement without any wildcard character ('*") always identifies a 'Set'. (A group of points measured with a tracker). Note that the first N points within the Set are taken as setup points (N = 2..6 - depending on shape type).
The matching capabilities through SQL statement are limited (only one statement at a time, limited exclusion…). Further, this wildcard method exclusively matches points (no shape origins, no shape history points)
However, this method might satisfy 90% of the needs. For all other cases, the data- file approach as described below must be used:
If a file, it - if not in the current directory - must be specified by a full path. It is simply a text file, each line taking exactly one point id of the following syntax '[P|S]:workpiece/pointID[#<devNr>]' (Ordinary points or Shape) or 'P:workpiece/setID:SetPtNr' (Set points)
'S' stands for 'Shape' i.e. indicates that this "point" relates to a shape origin. For ordinary points, 'P' must be used. devNr only must be specified if > 0! Default is 'no device number'. Do not specify '0' for "no device number". Omit the 'devNr' instead. For set points, the SetPtNr is mandatory.
Note that the separator for <devNr> has changed from ':' to '#' since Version 1.1.0. The colon ':' is now being used as separator for Set points. This change will not affect any existing scripts, but existing input data files for Shapefit calculations within existing scripts need to be adjusted. (However, very few people may have used this). See sample below for the syntax of such a data file.

Valid shape types:
2 = Line
4 = Plane
8 = Circle
16 = Cylinder

32 = Paraboloid
64 = Sphere
128 = Cone
If the given file is not found and if the passed string looks to be wildcard SQL statement, it is interpreted that way. This second option exclusively matches points (no shapes)

Return Value

Status	(Boolean)

Example

CoreM.**ShowAnalyzeShapefit** 4, “wp1/cir1“, False,
    C:\axyz\script\shpoints.dat“, False

where *shpoints.dat* may have the following contents:

P:Wp1/Set1:2
s:Plane/Plane2
P:Colani/Point1#3
p:Cone/Point1
S:Line/Line2

This might rather be a theoretical case. It takes one single set point (:2), two shape origins, one ordinary point and one device point (#3).
See also the sample scripts delivered with Axyz for other samples (mainly with SQL InputPoints argument)

2.6.14. ShowAnalyzeSinglePoint

Call the “Single Point” dialog box . The script language program flow will pause, until the dialog box is closed by the user.

Input Parameter

Point ID	(String)
Measurements	(String)
Comment	(String)

All parameters are optional. However, at least empty strings / zeros must be passed. See chapter 0 **Error! Unknown switch argument.** at the beginning of this section.
If the 'Measurements' parameter is an empty string, then all measurements related to the point are taken into account. Otherwise all measurements listed in the file are taken.
The 'Measurements' parameter relates either to a filename or to a wildcard SQL statement. This file - if not in the current directory - must be specified by a full path. It is simply a text file, each line taking exactly one point id of the following syntax 'workpiece/pointID/devicePtNr/Station'. For non- device points, the devicePtNr must always be zero.

Return Value

Status	(Boolean)
Example	CoreM.**ShowAnalyzeSinglePoint** "", "", ""

2.7. Help Menu Commands

2.7.1. ShowHelpAbout

Call the "About" dialog box. The script language program flow will pause, until the dialog box is closed by the user.

Return Value

Status	(Boolean)
Example	CoreM.**ShowHelpAbout**

2.7.2. ShowHelpContents

Call the "Help System" with is top level (Contents) page. The script language program flow will NOT pause. Thus a message box or something else to pause the script should follow this command.

Return Value

Status	(Boolean)
Example	CoreM.**ShowHelpContents**

2.7.3. ShowHelpTopic

Call the "Help System" of CDM and launch the specified topic page. The script language program flow will NOT pause. Thus a message box or something else to pause the script should follow this command.

Ask for a list with the current available help topics and their IDs.

Input Parameter

Topic ID	(Integer)

Return Value

Status	(Boolean)
Example	CoreM.**ShowHelpTopic 1**

3. CDM - Low Level Commands

3.1. General Remarks

This section describes the low level Core Module commands available through OLE automation. They can be used to generate scripts, where it is necessary to have more direct control over the Core Module.
Rule of thumb: Most of these functions also are available as ‘High Level’ version’. Just add “Show” in front of the command name to find the appropriate related function.
This will enable you to get additional information either through the description of the related high level function or by calling the related dialog box. Dialog boxes are often self explaining and if not, you even may call the related “Help”.
For that reason, the low level commands are not explained in very deep detail. However, the examples and the sample scripts should be documentation enough to allow you to write your own scripts. However, Basic Visual Basic knowledge is supposed.

Some functions need an argument to specify the Display Type for the Coordinates System. E.g. ShowSetAxisLabels().
These are valid values for **‘Coordinate Type’** Parameters:

1	**=**	**Rhr**	**(Right Handed Rectangular)**
2	**=**	**Lhr**	**(Left Handed Rectangular)**
3	**=**	**Ccl**	**(Cylindrical Clockwise)**
4	**=**	**Ccc**	**(Cylindrical Counterclockwise)**
5	**=**	**Scl**	**(Spherical Clockwise)**
6	**=**	**Scc**	**(Spherical Counterclockwise)**

All those functions that either include some printing or that do produce data to be stored (or both) have a parameter ‘Output Directive’ which can be specified as follows:

0	**=**	**Print Only**
1	**=**	**Save Only, allow Overwriting,**
2	**=**	**Save Only, no Overwriting**
3	**=**	**Print and Save, allow Overwriting,**
4	**=**	**Print and Save, no Overwriting**

If say ‘OutDirect’ was set to ‘2’ for the AnalysisIntersect function, it will always try to save the result. However, it an element with the specified ID already exists, the function will return with the appropriate error code. The Visual Basic program then could confirm the overwriting from the user and - depending on his answer - call the function again with ‘OutDirect’ set to ‘1’

All those functions that take overloaded Point/Shape IDs have so called **'SearchHint'** parameters. These tell the dB read function whether to take the Point- or the Shape Origin Coordinates to avoid potential ambiguity. Note that a 'Hint' parameter can incorporate hints for either one or for two elements. Rule: if the parameter is named nHint**s,** it refers to two elements, if it is called 'Hint' (without s), then it refers to one Element only. However, this meaning can easily be figured out according to the number of preceding Point Ids. Valid values for '**Search Hint**' Parameter:

(addressing two elements)

0	**=**	**Point, Point**
1	**=**	**Shape, Point**
2	**=**	**Point, Shape**
3	**=**	**Shape, Shape**

(addressing one element)

4	**=**	**Point, None**
5	**=**	**Shape, None**

Other types used are specific to one particular function and are described there.

Most functions return a Boolean value as their success status. The return status can be tested and if it is False, 'GetLastError()' can be used to retrieve the detailed error code.

The 'data retrieve' functions return the number of records found (0...numFound) on success and '-1' if an error has occurred. On '-1', use 'GetLastError()' to figure out the reason. (Note that '0' means that the query was successful, but none of the items matched the selection criteria - while -1 indicates a real error).

3.2. General Commands

3.2.1. GetLastError

Return the last error code.

<u>*Input Parameter*</u>
None
<u>*Return Value*</u>
Most recent Error Code (Long)

This function is typically called after any OLE function returned False. According to the returned code, the type of error can be figured out. See sample script for details.
This function replaces the 'errorStatus' property. Do no longer use this property.

List of Error Codes

11501	Invalid entry for first ID
11502	Invalid entry for second ID
11503	Twice the same element specified
11504	First element is not of appropriate type
11505	Second element is not of appropriate type
11506	Specified workpiece does not exist
11507	First point not found
11508	One or more parameter is missing (empty string)
11509	Specified file not found
11510	One of the parameters has inappropriate value
11511	Specified station not found
11512	Invalid Unit format
11513	Failed to open job (not existing or read-only)
11514	Twice the same element specified
11515	Invalid type specified
11516	Second point not found
11517	Third point not found
11518	Invalid entry for third ID
11519	Fatal internal error. Please send job file and description of the error to vendor
11520	error number currently unused
11521	Failed to save history information (shapefit- points) should never happen
11523	ID file has incorrect syntax
11524	error number currently unused
11525	Invalid entry for 4th ID
11526	Missing or invalid entry for one of the parameters (unknown which one)
11527	Invalid 'OutputDirective'
11528	Invalid 'Printer Options'
11529	Invalid 'Printer Page Length'
11530	Invalid coordinate system type
11531	Invalid coordinates for far/near rod point
11532	error number currently unused
11533	Missing history information
11534	Device not found
11535	IDs of device and device points do not match.
11536	IDs of device and device points do not match in terms of count.
11537	Inappropriate search-hint
11538	Invalid shape type specified
11539	Invalid calculation type specified
11540	Failed to read setup points (fatal - should never happen. Please send job to Leica)
11541	Alpha offset processing failed; Check your 'Remove/Add' portions!
11542	One of the parameters is not correct (don't know which one).
11543	Feature not yet implemented
11544	Failed to store focus point (applies to parabola shapefit)
11545	Invalid approximation method specified
11546	Specified shape not found
11547	Invalid scale or unit factor; It is zero (danger of divison!)
11553	Refused to save due excluded measurements; Not all measurements of this point have been included. For consistency reasons, measurements must be excluded at the Data Manager level (Delete them or set them as 'Not Used').
11560	Invalid 'Thickness Type' specified
11561	Thickness correction failed for unknown numerical reasons; Check all your thickness info!
11562	Shape on which correction is based does not exist
11563	Invalid ID specified for shape on which correction is based
11564	Amount of thickness is zero. (Such a correction would have no effect - so remove thickness CS ID to disable correction.)
11565	Invalid thickness type
11566	If thickness correction AND reflector correction both are applied, then the 'Planar Flag' must be the same for both.
11567	If thickness correction AND reflector correction both are applied, then the 'Correction CS' must be the same for both.

```
11570    Invalid 'Reflector offset type' specified
11571    Specified reflector not found
11572    Reflector correction failed for unknown numerical reasons;
         Check all your reflector offset info!
11573    Amount of reflector offset is zero (such a correction would
         have no effect - so remove reflector CS ID to disable
         correction)
11574    Invalid reflector offset type
11575    Invalid reflector ID
11576    Shape on which reflector correction is based does not exist
11577    Invalid ID specified for shape on which reflector correction
         is based
11580    Calculation failed for unknown reason
```

Example

if SetWorkpiece(“wing1“, False) = False then
  nStatus = **GetLastError**(); // could be printed out
endif

3.2.2. *IsMeanErrorMode*

Get the default error mode according to general settings.

Return Value
(Boolean)
i.e. True if system is in ‘Standard deviation’ mode,
False if system is in ‘Uniform weigthing (RMS)’
Example
bIsMean = CoreM.**IsMeanErrorMode**()

3.2.3. *CallTask*

Call a task (Equivalent to the Visual Basic “Shell” command).

This function does NOT pause the script. If you like to have your script paused, you may call the Visual Basic ‘Sleep’ Function. An Alternative to this Command is the Visual Basic Function ‘Create Process’. You will find in the Declaration section of the “Show Form on top.Bas” file, which is located in the *Scripts* subdirectory of the ***Axyz*** data directory.

Input Parameter

Command Line	(String)

Return Value

True, if command executed successfully	(Boolean)

Example CoreM.**CallTask** “C:\windows\notepad.exe file.txt“

3.2.4. IsTaskRunning

Checks if a task is already running, given by its executable name.

It is not recommended to use this command to start the theodolite manager STM/MTM, as the script will not pause until the initialisation of the program is finished.

<u>*Input Parameter*</u>

Executable name of the task	(String)

<u>*Return Value*</u>

True if task is running	(Boolean)

<u>*Example*</u>

```
if (CoreM.IsTaskRunning(“calc.exe“)) then
   CoreM.CallTask “calc.exe“
endif
```

3.3. Settings Menu Commands

3.3.1. SetWorkpiece

Set the ‘current workpiece’ according to the specified ID. If the workpiece specified does not exits, the function will return False unless the second parameter is set to ‘True’, which will cause the WP ID to be added.

<u>*Input Parameter*</u>

New current workpiece ID	(String)
Add if not exists flag	(Boolean

<u>*Return Value*</u>

Status	(Boolean)

<u>*Example*</u>

```
bStatus = CoreM.SetWorkpiece(“wing1“, False);
```

3.3.2. SetUnits

Set new current units.

<u>*Input Parameter*</u>

Units Type	(Integer)
New current unit of the specified type	(String)
Format	(Integer)

Units type can be one of the following values. Note that only one of these types can be set upon one function call.

1 = length
2 = angle
3 = temperature
4 = pressure
The ‘current unit’ names depend on how specified in the database. (language dependent)

‘Format’ indicates the number of decimals. In the special case where angle is ‘degree’, Format can also be specified as ‘-1’. This sets the format to ‘sexagesimal’.

Return Value
Status (Boolean)
Example bStatus = CoreM.**SetUnits**(1, “meters“, 5)
bStatus = CoreM.**SetUnits**(2, “degree“, -1);

3.3.3. SetWarnings

Set new current analysis warning values.

Input Parameter

Item Type	(Integer)
New value of the specified item in [m]	(Double)

The item type can take one of the following values. Note that only one of these types can be set upon one function call.
0 = parallel Tolerance
1 = intersect Tolerance
2 = X Max Tol
3 = Y Max Tol
4 = Z Max Tol
5 = Total Max Tol (X,Y,Z)
6 = Temp Max Tol

Return Value
Status (Boolean)
Example CoreM.**SetWarnings** 1, 0.001
i.e. set intersection Tolerance to 1 millimeter

3.3.4. SetLogfileOutput

Set new in progress printing parameter for the logfile.

<u>*Input Parameter*</u>

Print Level	(Integer)
Page Length	(Integer)
Page Width	(Integer)
Logfile Name	(String)
‘Append’ flag	(Boolean)

The ‘Print Level’ parameter can take one of the following values:
1 = None
2 = Reduced
4 = Full
8 = Include commands printing
16 = Print on request only

The first three values 1..4 are exclusive. However the latter two can be added together and to one of the first three.

Example for valid values:
4 : Full Output on each calculation
8 : Print commands only
9 (=1+8) : print commands only
26(=2+8+16) : print reduced output and commands, but on request only
12(=4+8) : print full output and commands on each calculation
17: Does not make sense
Note that ‘Print on request’ affects ‘Reduced’ and ‘Full’ only - it is independent from ‘Commands’ printing

<u>*Return Value*</u>

Status	(Boolean)

<u>*Example*</u>
bStatus = CoreM.**SetLogfileOutput**(26, 60, 80, “logfile.log“, False)
i.e. enable Reduced printout to *logfile.log*, including commands, but on request only.

3.3.5. SetPrintOutput

Set new in progress printing parameter for the printer.

<u>*Input Parameter*</u>

Print Level	(Integer)

Page Length	(Integer)
Page Width	(Integer)

For valid values of ‘Print Level’, see command ‘SetLogfileOutput’

<u>*Return Value*</u>

Status (Boolean)

<u>*Example*</u>

bStatus = CoreM.**SetPrintOutput**(4, 60, 80)

i.e. enable Full, not including commands, automatic printout on each calculation, page length 60 lines, page width 80 columns)

3.3.6. SetOverwriteConfirms

Set current overwrite confirm flags.

<u>*Input Parameter*</u>

Confirm flags (Integer)

Confirm flags can be composed as the sum of the following values:

1 = ORIENTATION
2 = MEASUREMENT
4 = ELEMENT
8 = FILE
16 = CONTROLPOINT
32 = ENTEREDPOINT
64 = CALCPOINT
128 = MEASUREDPOINT
256 = HIDDENPOINT

e.g. 74 (=2+8+64) asks for overwrite confirmation of Measurements, Files and Calculated points.

<u>*Return Value*</u>

Status (Boolean)

<u>*Example*</u> bStatus = CoreM.**SetOverwriteConfirms**(128)

3.3.7. SetDeleteConfirms

Set current delete confirm flags.

<u>*Input Parameter*</u>

Confirm flags (Integer)

For valid confirm flags, see chapter 0 **Error! Unknown switch argument.** command.

Return Value

Status	(Boolean)
Example	bStatus = CoreM.**SetDeleteConfirms**(106)

3.3.8. SetLabelAxis

Set current axis labels.

Input Parameter

Coordinate system type	(Integer)
New Label for X Axis	(String)
New Label for Y Axis	(String)
New Label for Z Axis	(String)

For valid coordinate system types: see top of this section

Return Value

Status	(Boolean)
Example	bStatus = CoreM.**SetLabelAxis(**2 “&X“, “&Y“, “&Z“)

i.e. set these labels to type ‘Lhr’. ‘&’ marks the following character as ‘mnemonic’

3.3.9. SetGeneral

Set general global system flags.

Input Parameter

Item type	(Integer)
Enable/Disable	(Boolean)
Format for ‘Vector Display’	(Integer)

Item type can be one of the following values. Note that only one of these types can be set upon one function call

0 = NormalVector,
2 = SexGesDegree,
3 = RadiusView,
4 = AutoCalculate,
5 = OpenRecentJob,
6 = MeanErrMode,
7 = ShowLocErrSwitch,

The ‘Format’ parameter is ignored except for ‘Vector display’.

Return Value	
Status	(Boolean)
Example	bStatus = CoreM.**SetGeneral**(1, False, 0)
	CoreM.**SetGeneral** 3, True, 8;

3.4. Tools Menu Commands

3.4.1. MovePoints

Calls the ‘Swap Points’ function.

Input Parameter	
Single point ID or SQL point ID	(String)
Reference Workpiece ID	(String)
Workpiece ID	(String)
Coordinate system ID for reference points	(String)
Copy source points (if true), or move (if false)	(Boolean)
Overwrite existing destination points (if true)	(Boolean)
Return Value	
Status	(Boolean)

Example

CoreM.**MovePoints** “wp1/p*”, “refwp1”, “wp1”, “default/base”, False, True

3.5. Coordinate System Menu Commands

3.5.1. ActiveCoordSys

Set the ‘current coordinate system according to the specified ID. If the workpiece specified does not exits, the function will return False.

Input Parameter

New current coordinate system ID (String)

Return Value	
Status	(Boolean)
Example	bStatus = CoreM.**ActiveCoordSys**(“default/base“);

3.5.2. CoordSystemType

Set current display type for the coordinate system.

Input Parameter

Coordinate system type (Integer)

For valid coordinate system types, see chapter 0**Error! Unknown switch argument.**.

Return Value

Status	(Boolean)
Example	bStatus = CoreM.**CoordSystemType** 6

i.e. set coord system type to SCC. i.e. 'spherical counter- clockwise.

3.5.3. ScaleCoordSys

Create a new coordinate system by scaling an existing one. The scale factor can either be specified directly, or if set to 0.0, it will be calculated from two points and a distance (which must not be 0 in that case).

Input Parameter

Starting Coordinate System ID	(String)
New Coordinate System ID	(String)
Scale Factor	(Double)
First auxiliary Point ID	(String)
Second auxiliary Point ID	(String)
Search Hints for the two points above	(Integer)
Distance between auxiliary Points	(Double)
Comment	(String)
Output directive	(Integer)

For valid values of 'Search Hints' and 'Output directive', see chapter 0 ***Error! Unknown switch argument.***.

Return Value

Status (Boolean)

Example

bStatus = CoreM.**ScaleCoordSys**("default/sc2", "default/sc4", 2.0, "",
"",0, 0., "", 3)

i.e. scale sc2 by factor 2.0 and store/ overwrite result under sc4, also print.

CoreM.**ScaleCoordSys** "default/xy", "wp2/scx", 0.0, "default/cir1", "default/cir2",3, 2.5, "comment", 0

i.e. scale xy by factor calculated from two circle origins that have the given distance 2.5 meters. and store result under sc4, do not store, but print result!

3.5.4. RotateCoordSys

Create a new coordinate system by rotating an existing one. The rotation is either about an axis or about an axis parallel in the distance of a given point.

Input Parameter

Starting Coordinate System ID	(String)
New Coordinate System ID	(String)
Rotation Angle	(Double)
Rotation Axis	(Integer)
Point ID to rotate about	(String)
Search Hints for the point above	(Integer)
Comment	(String)
Output directive	(Integer)

The rotation angle must be in radian. For valid values of ‘Search Hints’ and ‘Output directive’, see chapter 0 ***Error! Unknown switch argument.***

The ‘Rotation Axis’ parameter can take one of the following values:
1 = X
2 = Y
3 = Z

Return Value

Status	(Boolean)

Example

bStatus = CoreM.**RotateCoordSys**(“default/rt1“, “default/rt2“, 0.321,
1”“,0, ““, 3)

i.e. rotate rt1 about X by 0.321 rad and store result in rt2.

CoreM.**RotateCoordSys** “default/rt1“, “wp2/rtnew“, 0.5, 5”default/cir1“, ,3, “comment“, 2

i.e. rotate rt1 by angle 0.5 radians about circle cir1 origin, axis is Z. Save but do not overwrite in case of exists, do not print.

3.5.5. TranslateCoordSys

Create a new coordinate system by translating an existing one. The translation components can either be specified directly, or a point to which to translate ca be specified.

Input Parameter

Starting Coordinate System ID	(String)

New Coordinate System ID	(String)
X Translation Component	(Double)
Y Translation Component	(Double)
Z Translation Component	(Double)
Point ID to translate to	(String)
Search Hints for the point above	(Integer)
Comment	(String)
Output directive	(Integer)

The translation components must be in meters. For valid values of 'Search Hints' and 'Output directive', see chapter 0 ***Error! Unknown switch argument.***.

Return Value

Status	(Boolean)

Example

bStatus = CoreM.**TranslateCoordSys**("default/t1", "default/t2", 1., 1. 1.,"", 0, "comment", 3)

i.e. translate by (1,1,1) meters.

CoreM.**TranslateCoordSys** "default/t1", "wp2/tnew", 0., 0., 0. "default/cir1", ,5, "comment", 0

i.e. translate to origin of circle cir1, only print result.

3.5.6. TransformCoordSys

Perform a 'least squares transformation'. Note that the 'update' feature is not available within this function.

Input Parameter

New Coordinate System ID	(String)
Calculate Scale Flag	(Boolean)
Input Points	(String)
Existing CS ('for approximation')	(String)
Mean Error Flag	(Boolean)
Prefix to remove	(String)
Prefix to add	(String)
Add to Point Number	(Long)
nIterationSteps	(Integer)
percentage change	(Double)
Comment	(String)
Output directive	(Integer)

For valid values of ‘Search Hints’ and ‘Output directive’, see chapter 0 ***Error! Unknown switch argument.***.
If Existing Coordinate System is an empty string, the first three points of the reference points are taken as setup points. If an existing system is specified, then this serves as the initial approximation.

The ‘Input Points’ parameter relates either to a filename or to a wildcard SQL statement. This file - if not in the current directory - must be specified by a full path. It is simply a text file, each line taking exactly one point id of the following syntax ‘refID/workpiece/pointID’.

Return Value

Status	(Boolean)

Example

CoreM.**TransformCoordSys** “default/ct1“, False, “refWing”, ““, False, “refWing/wp1”,”default“, 0, 2, 1#, “comment“, 3

i.e. take reference points from ‘refWing/wp1’ and match with points from ‘default’. Calculate using RMS method, take first three points of ‘refWing’ as setup points

3.5.7. AxisAlignCoordSys

Create a new coordinate system by ‘Axis Alignment’.

Input Parameter

New Coordinate System ID	(String)
Align Point 1	(String)
Control Point Coord X	(Double)
Control Point Coord Y	(Double)
Control Point Coord Z	(Double)
Align Point 2	(String)
Search Hints for Point1 and Point 2	(Integer)
Align Type2	(Integer)
Align Point 3	(String)
Search Hint for Point3	(Integer)
Align Type3	(Integer)
Horizontal Plane Flag	(Boolean)
Comment	(String)
Output directive	(Integer)

For valid values of ‘Search Hints’ and ‘Output directive’, see chapter 0 ***Error! Unknown switch argument.***.

The ‘Align Type’ parameter can take the following values:
1 = X
2 = Y
3 = Z
-1 = -X
-2 = -Y
-3 = -Z

Return Value

Status	(Boolean)

Example

bStatus = CoreM.**AxisAlignCoordSys(**“wp1/ax1“, “wp1/pt1“, 0., 0., 0., “wp1/pt2“,0, 3, “wp1/pt3“, 4 -2, False, “This is a comment“, 1)

3.5.8. *AxisAlignCoordSysEx*

Create a new coordinate system by ‘Axis Alignment’. This function is an expansion of ‘AxisAlignCoordSys’ with 6 new parameters. Note that 3 parameters have been renamed (dControlX, -Y and –Z). Otherwise the functions are identic. The new parameters are offset values. See the appropriate dialog in the CDM’s function ‘Axis Align’.

Input Parameter

New Coordinate System ID	(String)
Align Point 1	(String)
dControlX	(Double)
dControlY	(Double)
dControlZ	(Double)
dOffset1P1	(Double)
dOffset2P1	(Double)
dOffset3P1	(Double)
Align Point 2	(String)
Search Hints for Point1 and Point 2	(Integer)
Align Type2	(Integer)
dOffset1P2	(Double)
dOffset2P2	(Double)
Align Point 3	(String)
Search Hint for Point3	(Integer)
Align Type3	(Integer)
Horizontal Plane Flag	(Boolean)
DOffset1P3	(Double)
Comment	(String)

Output directive	(Integer)

Return Value

Status	(Boolean)

Example

bStatus = CoreM.**AxisAlignCoordSysEx(**“wp1/ax1“, “wp1/pt1“, 0., 0., 0., 1., 1., 2., “wp1/pt2“,0, 3, 2., 2., “wp1/pt3“, 4 -2, False, 3., “This is a comment“, 1)

(This statement must be on same line in code)

3.6. Analysis Menu Commands

3.6.1. AnalyzeParallel

Calculates a new Line/Plane parallel to an existing Shape Axis/Plane.

Input Parameter

Calculation Type	(Integer)
Element 1	(String)
Element 2	(String)
Search Hint for Element 2	(Integer)
New Shape ID	(String)
Output direction	(Integer)

For valid values of ‘Search Hints’ and ‘Output directive’, see chapter 0 ***Error! Unknown switch argument.***.
Search hint is only of interest it the second Element is a point not a line.

The ‘Calculation Type’ parameter can take the following values:
0 = ParallelLine
1 = ParallelPlane

Return Value

Status	(Boolean)

Example

bStatus = CoreM.**AnalyzeParallel**(1, “wp1/shape1“, “wp1/shape2“, 1, “wp1/newln“, 3)

3.6.2. AnalyzePerpend

Calculates a new perpendicular Line/Plane to an existing Shape Axis/Plane.

Input Parameter

Calculation Type	(Integer)
Element 1	(String)
Element 2	(String)
Search Hint for Element 2	(Integer)
New Shape ID	(String)
Output direction	(Integer)

For valid values of 'Search Hints' and 'Output directive', see chapter 0 ***Error! Unknown switch argument.***.
Search hint is only of interest it the second Element is a point not a line.

The 'Calculation Type' parameter can take the following values:
0 = PerpendPtShAxis
1 = PerpendPtShSurf
2 = PerpendShAxisShAxis

Return Value

Status	(Boolean)

Example

bStatus = CoreM.**AnalyzePerpend**(2, "wp1/shape1", "wp1/shape2", 1, "wp1/newsh", 1)

3.6.3. AnalyzeBisect

Calculates a bisecting element given two elements.

Input Parameter

Calculation Type	(Integer)
Element 1	(String)
Element 2	(String)
Search Hints for Elements1/2	(Integer)
New Element ID	(String)
Output direction	(Integer)

For valid values of 'Search Hints' and 'Output directive', see chapter 0 ***Error! Unknown switch argument.***.

The 'Calculation Type' parameter can take the following values:
0 = BisectPtPt
1 = BisectPtShAxis
2 = BisectPtPlane
3 = BisectShAxisShAxis

4 = BisectShAxisPlane
5 = BisectPlanePlane

Return Value

Status	(Boolean)

Example

bstatus = CoreM.**AnalyzeBisect**(1, “w1/pt1“, “w1/pt2“, 1, “w1/new“, 3)

3.6.4. AnalyzeProjection

Project a point (normal) to a shape surface. In case of a circle, the projection applies to the circle line (not the circle plane!)

Input Parameter

Point to project	(String)
Shape	(String)
Search Hints for Point	(Integer)
New Point ID1	(String)
Output direction	(Integer)

For valid values of ‘Search Hints’ and ‘Output directive’, see chapter 0 ***Error! Unknown switch argument.***.

Return Value

Status	(Boolean)

Example

bstatus = CoreM.**AnalyzeProjection**(“w1/pt1“, “w1/plane1“, 2,
“w1/projpt“, 3)

3.6.5. AnalyzeIntersect

Calculates a intersecting element given two shapes.

Input Parameter

Calculation Type	(Integer)
Element 1	(String)
Element 2	(String)
New Element ID1	(String)
New Element ID2	(String)
Output Directive	(Integer)

In some cases there result two new elements (e.g. intersection of a sphere and line)
For valid values of ‘Search Hints’ and ‘Output directive’, see chapter 0 ***Error! Unknown switch argument.***.

The ‘Calculation Type’ parameter can take the following values:
0 = InterShAxisShAxis
1 = InterShAxisShSurf
2 = InterShSurfShSurf

<u>*Return Value*</u>

Status	(Boolean)

<u>*Example*</u>

bStatus = CoreM.**AnalyzeIntersect**(1, “wp1/sh11“, “wp1/sh22“, “default/sh33“, ““, 3)

3.6.6. AnalyzeDivideLine

Calculate line division points.

<u>*Input Parameter*</u>

Element 1	(String)
Element 2	(String)
Search Hints for Element1/2	(Integer)
Number of Division points	(Integer)
Starting ID of created points	(String)
Offset distance from first point	(Double)
Output Directive	(Integer)

For valid values of ‘Search Hints’ and ‘Output directive’, see chapter 0 ***Error! Unknown switch argument.***.

<u>*Return Value*</u>

Status	(Boolean)

<u>*Example*</u>

bStatus = CoreM.**AnalyzeDivideLine** “wp1/pt1“, “wp1/pt2“, 4, 1, “dp“, 0. 0, 3

3.6.7. AnalyzeCompareSets

Perform a ‘Set Compare’ between reference and other points.

⚠ This function is not yet implemented.

Input Parameter

Input Points	(String)
Reference coordinate system ID	(String)
Prefix to remove	(String)
Prefix to add	(String)
Point number to add	(Long)
'Show out of tolerance only' flag	(Boolean)
'Tolerances from reference' flag	(Boolean)
Output Directive	(Integer)

Return Value

Status	(Boolean)
Example	---

3.6.8. AnalyzeTwoPoint

Performs a 'two point analysis' on two given points.

Input Parameter

Element 1	(String)
Element 2	(String)
Search Hints for Elements1/2	(Integer)

This function does neither produce an element to be stored nor does it show its results in a dialog. Thus the calling of this function without printing the results does not make sense. The results are printed by default according to the settings of 'Online Output' settings. Make sure that these are not set to 'no output' for both, logfile and printer.
For valid values of 'Search Hints', see chapter 0 ***Error! Unknown switch argument.***.

Return Value

Status (Boolean)

Example

bStatus = CoreM.**AnalyzeTwoPoint**("wp1/pt1", "wp1/pt2", 3)

3.6.9. AnalyzeAngle

Perform an angle calculation.

Input Parameter

Input Type1	(Integer)
Element 11	(String)
Element 12	(String)
Search Hints11/12	(Integer)
Input Type2	(Integer)
Element 21	(String)
Element 22	(String)
Search Hints21/22	(Integer)

The ‘Input Type’ parameters can take the following values:
0 = AngTwoPoint
1 = AngShapeAxis
2 = AngVector

Depending on the types, Element12/Element22 become obsolete (if element 11/21 is either a vector or a shape axis, i.e. not a point)

Return Value

Status	(Boolean)

Example

bOK = CoreM.**AnalyzeAngle**(0, “wp1/pt1“, “wp1/pt1“, 2, 0, “wp2/pt1“, “wp2/pt1“, 2)

3.6.10. AnalyzeDistance

Perform a ‘Distance’ calculation.

Input Parameter

Calculation Type	(Integer)
Element 1	(String)
Element 2	(String)
Search Hints1/2	(Integer)

This function does neither produce an element to be stored nor does it show its results in a dialog. Thus the calling of this function without printing the results does not make sense. The results are printed by default according to the settings of ‘Online Output’ settings. Make sure that these are not set to ‘no output’ for both, logfile and printer.
For valid values of ‘Search Hints’, see chapter 0 ***Error! Unknown switch argument.***.

The ‘Calculation Type’ parameter can take the following values:
0 = DistPtPt
1 = DistPtShAxis
2 = DistPtShSurf
3 = DistShAxisShAxis
4 = DistPlanePlane
5 = DistShAxisPlane

Return Value

Status	(Boolean)

Example

bStatus = CoreM.**AnalyzeDistance**(0, “wp1/pt1“, “wp1/pt2“, 0)

3.6.11. AnalyzeArcDistance

Perform an ‘Arc Distance’ calculation.

Input Parameter

Element 1	(String)
Element 2	(String)
Search Hints1/2	(Integer)
Arc Radius	(Double)

This function does neither produce an element to be stored nor does it show its results in a dialog. Thus the calling of this function without printing the results does not make sense. The results are printed by default according to the settings of ‘Online Output’ settings. Make sure that these are not set to ‘no output’ for both, logfile and printer.
For valid values of ‘Search Hints’, see chapter 0 ***Error! Unknown switch argument.***.

Return Value

Status	(Boolean)

Example

bStatus = CoreM.**AnalyzeArcDistance** “wp1/pt1“, “wp1/pt2“, 2, 100.0

3.6.12. AnalyzeShapefit

Perform a ‘Shape Bestfit’ calculation.

Input Parameter

Shape type	(Integer)
Shape ID	(String)
Thickness Correct Flag	(Boolean)
Input Points	(String)
Mean Error Flag	(Boolean)
Origin Point ID	(String)
Vector ID	(String)
Minimal iteration steps	(Integer)
Percent change criteria	(Double)
Comment	(String)
Output Directive	(Integer)

Shape type is mandatory. The Shape ID is optional as long no save is requested (Note that printing will not be able to print a ID name). Origin Point ID and Vector ID are both optional (i.e. you may pass empty strings). These parameters indicate whether the to store the origin point and/or the vector (see Shapefit solution dialog). Minimal iteration steps and Percent change criteria allow to control the termination criteria. Reasonable defaults are ‘10’ for the former and ‘0.1%’ for the latter.

Valid shape types:
2 = Line
4 = Plane
8 = Circle
16 = Cylinder
32 = Paraboloid
64 = Sphere
128 = Cone

The ‘Input Points’ parameter relates either to a filename or to a wildcard SQL statement.
If a SQL statement, it has typically a syntax like ‘workpiece1/plane*’ (or ‘workpiece1/plane*/*’ in order to include also non-zero device points - Note that ‘workpiece1/plane*’ is equivalent to ‘workpiece1/plane*/0’). This means to select all points whose name starts with ‘plane’ (plane1, plane2, plane3…) within the workpiece1.
A statement without any wildcard character (‘*”) always identifies a ‘Set’. (A group of points measured with a tracker). Note that the first N points within the Set are taken as setup points (N = 2..6 - depending on shape type).
The matching capabilities through SQL statement are limited (only one statement at a time, limited exclusion…). Further, this wildcard method exclusively matches points (no shape origins, no shape history points)

However, this method might satisfy 90% of the needs. For all other cases, the data- file approach as described below must be used:
If a file, it - if not in the current directory - must be specified by a full path. It is simply a text file, each line taking exactly one point id of the following syntax '[P|S]:workpiece/pointID[#<devNr>]' (Ordinary points or Shape) or 'P:workpiece/setID:SetPtNr' (Set points)
'S' stands for 'Shape' i.e. indicates that this "point" relates to a shape origin. For ordinary points, 'P' must be used. devNr only must be specified if > 0! Default is 'no device number'. Do not specify '0' for "no device number". Omit the 'devNr' instead. For set points, the SetPtNr is mandatory.
Note that the separator for <devNr> has changed from ':' to '#' since Version 1.1.0. The colon ':' is now being used as separator for Set points. This change will not affect any existing scripts, but existing input data files for Shapefit calculations within existing scripts need to be adjusted. (However, very few people may have used this). See sample below for the syntax of such a data file.

<u>*Return Value*</u>

Status	(Boolean)

<u>*Example*</u>
bStatus = CoreM.**AnalyzeShapefit(**4, "wp1/cir1", False,
"C:\axyz\script\shpoints.dat", False, "", "", 3, 0.1, "", 3)
where *shpoints.dat* may have the following contents:

P:Wp1/Set1:2
s:Plane/Plane2
P:Colani/Point1#3
p:Cone/Point1
S:Line/Line2

This might rather be a theoretical case. It takes one single set point (:2), two shape origins, one ordinary point and one device point (#3).
See also the sample scripts delivered with Axyz for other samples (mainly with SQL InputPoints argument)

3.6.13. AnalyzeSinglePoint

Perform a 'Single Point' calculation. This feature can typically be used to recalculate previously points using oriented stations.

<u>*Input Parameter*</u>

Point ID	(String)
Measurements	(String)
Result Point ID	(String)

Comment	(String)
Output Directive	(Integer)

If the ‘Measurements’ parameter is an empty string, then all measurements related to the point are taken into account. Otherwise all measurements listed in the file are taken.
The ‘Measurements’ parameter relates either to a filename or to a wildcard SQL statement. This file - if not in the current directory - must be specified by a full path. It is simply a text file, each line taking exactly one point id of the following syntax ‘workpiece/pointID/devicePtNr/Station’. For non- device points, the devicePtNr is always zero.
For valid values of ‘Output Directive’, see chapter 0 ***Error! Unknown switch argument.***.

Return Value

Status	(Boolean)

Example

bStatus = CoreM.**AnalyzeSinglePoint(**“wp1/pt1“, “” “wp1/ptnew1“, “”, 3)

3.6.14. *AnalyzeHiddenPoint*

Perform a ‘Hidden Point’ calculation.

Input Parameter

Hidden Point to calculate	(String)
Setup points flags	(Integer)
Comment	(String)
Output Directive	(Integer)

Return Value

Status	(Boolean)

The ‘setup points flag’ can be used to take control over which points of the device are taken as setup points. If 0, the first three are taken. Use this value as a bit map to define the nr of point in the list as setup point, e.g. 21 (1 + 4 + 16) means: point 1, point 2 and point 4 are setup points.

For valid values of ‘Output Directive’, see chapter 0 ***Error! Unknown switch argument.***.

Example

bOK = CoreM.**AnalyzeHiddenPoint**(“wp1/HP1“, 0, “comment“, 3)

3.6.15. ExAnalyzeHiddenPoint

Perform a ‘Hidden Point’ calculation with the option to specify an offset to the tip point. This function is the same as described in 0 ***Error! Unknown switch argument.*** except that there are 3 extra parameters.

Input Parameter

Hidden Point to calculate	(String)
X Offset	(Double)
Y Offset	(Double)
Z Offset	(Double)
Setup points flags	(Integer)
Comment	(String)
Output Directive	(Integer)

Return Value

Status	(Boolean)

See chapter 0 ***Error! Unknown switch argument.*** for further details.

3.6.16. AnalyzeNominalShapefit

Perform a ‘Nominal Shape Bestfit’ calculation. This function uses parameters instead of setup points. Note that the interface is the same as for “AnalyzeShapet” except that “AnalyzeNominalShapefit” has 8 additional parameters (marked with asterix). Refer to the descrption of “AnalyzeShapefit” for other parameters.

Input Parameter

Shape type	(Integer)	
Shape ID	(String)	
Nominal Origin X	(Double)	*
Nominal Origin Y	(Double)	*
Nominal Origin Z	(Double)	*
Nominal Rotation X	(Double)	*
Nominal Rotation Y	(Double)	*
Nominal Rotation Z	(Double)	*
Nominal Size	(Double)	*
Fixed Flags	(Integer)	*

Thickness Correct Flag	(Boolean)
Input Points	(String)
Mean Error Flag	(Boolean)
Origin Point ID	(String)
Vector ID	(String)
Minimal iteration steps	(Integer)
Percent change criteria	(Double)
Comment	(String)
Output Directive	(Integer)

"Fixed Flags" is a number composed on binomial values for fixed components. Values may be added (ored):

Nominal Origin X fixed	= 1
Nominal Origin Y fixed	= 2
Nominal Origin Z fixed	= 4
Nominal Rotation X fixed	= 8
Nominal Rotation X fixed	= 16
Nominal Rotation X fixed	= 32
Size Parameter fixed	= 64

If all 7 parameters are fixed (value = 127), the parameter "Input Points" is not used. This creates a shape with all 7 parameters given.

3.7. Updating commands

3.7.1. UpdatePoint

Perform a 'Single Point update', i.e. recalculate an existing point.

Input Parameter

Point ID	(String)
Comment	(String)
Output Directive	(Integer)

It is assumed that the point with the ID passed as first parameter already exists. If comment specified and option to save, then the original comment will be replace by the new one. Depending on Output Directive, you also may update a point without having to overwrite the exiting one but doing a printout only.
For valid values of 'Output Directive', see chapter 0 ***Error! Unknown switch argument.***.

Return Value

Status	(Boolean)

Example

CoreM.**UpdatePoint** “wp1/pt1“, “point updated 12.4.96”, 3

3.7.2. UpdateAllPoints

Perform a ‘Point update’ for all points, i.e. recalculate all existing points.

Input Parameter

Comment	(String)
Ignore Errors	(Boolean)
Output Directive	(Integer)

If comment specified and option to save, then the original comments of all points will be replaced by the new one. Depending on Output Directive, you also may update points without having to overwrite the exiting one but doing a printout only.
If the ‘Ignore Errors’ flag is True, the process will not stop if something will fail (e.g. if the measurements or station for a particular point are not found).

Return Value

Status	(Boolean)

Example

CoreM.**UpdateAllPoints** “updated 12.3.96“, TRUE, 1

3.8 New CDM Low Level commands since Axyz V1.4.0 Service Pack #1

See further remarks at the bottom of this section

3.8.1. Listing of new Functions:

BOOL GetOverwriteConfirms(short* punFlags);

BOOL GetDeleteConfirms(short* punFlags);

```
BOOL GetWarnings(short nItem, double* pdValue);

BOOL GetUnits(short nType, BSTR* lpszUnit, short* pnFormat);

BOOL GetGeneral(short nType, BOOL* pbEnable, short* nFormat);

BOOL SetActiveWorkpiece(LPCSTR lpszWpId,
                        BOOL bAddIfNotExists);

BOOL GetActiveWorkpiece(BSTR* lpszWpId);

BOOL SetActiveCoordSys(LPCTSTR lpszCSysName);

BOOL GetActiveCoordSys(BSTR* lpszCSysName);

BOOL AnalyzeTwoPointEx(LPCSTR lpszElemId1,
                        LPCSTR lpszElemId2,
                        short  nSearchHints,
                        BOOL bDoPrint,
                        double* pdVX,
                        double* pdVY,
                        double* pdVZ,
                        double* pdDistance);

BOOL AnalyzeAngleEx(short nFromType,
                    LPCSTR lpszElemId11,
                    LPCSTR lpszElemId12,
                    short nFromSearchHints,
                    short nToType,
                    LPCSTR lpszElemId21,
                    LPCSTR lpszElemId22,
                    short nToSearchHints,
                    BOOL bDoPrint,
                    double* pdAngle);

BOOL AnalyzeDistanceEx(short nType, LPCSTR lpszFrom,
                        LPCSTR lpszTo,
                        short nSearchHints, BOOL bDoPrint,
                        double* pdDistance);

BOOL AnalyzeArcDistanceEx(LPCSTR lpszElemId1,
                        LPCSTR lpszElemId2,
                        short nSearchHints, double dRadius,
                        BOOL bDoPrint, double* pdChordDist,
                        double* pdArcDist,
                        double* pdSubtendedAngle);

BOOL AnalyzeIntersectEx(short nType, LPCSTR lpszElemId1,
                              LPCSTR lpszElemId2,  LPCSTR lpszResultId,
                              LPCSTR lpszResId2, short  nOutDirect,
                              double* pdDistance, double* pdAngle);

BOOL AnalyzePerpendEx(short nType, LPCSTR lpszElemId1,
                             LPCSTR lpszElemId2,
                             short  nSearchHints, LPCSTR lpszResultId,
                             short nOutDirect, double* pdDistance);

BOOL AnalyzeBisectEx(short nType, LPCSTR lpszElemId1,
                          LPCSTR lpszElemId2, short nSearchHints,
                          LPCSTR lpszResultId,
                          short nOutDirect, double* pdDistance);

BOOL AnalyzeProjectionEx(LPCSTR lpszPoint, LPCSTR lpszShape,
                                short nSearchHints, LPCSTR lpszResult,
                                short nOutDirect, double* pdDistance);

BOOL NormalPlaneFromLine(LPCSTR lpszLineID,
                        LPCSTR lpszResultPlaneID,
                        LPCSTR lpszThroughPointID);
```

3.8.2. Description and Remarks

'Get' Functions are the opposite of (mostly already existing) 'Set' functions. See documentation of 'Set' functions for details (meaning of unFlags, nType, nItem..
Note that the notation of the parameter types slightly differs from the one usually used witin this document.

SetActiveWorkpiece(): Same function as SetWorkpiece(). Added for name consistency reasons. Old function SetWorkpiece() still available for compatibility reasons. However, new projects should use SetActiveWorkpiece()

SetActiveCoordSys(): Same function as ActiveCoordSys(). Added for name consistency reasons. Old function ActiveCoordSys() still available for compatibility reasons. However, new projects should use SetActiveCoordSys()

Note: If using BSTR output parameters (for example 'GetActiveCoordSys(&str)') from a C++ client, it is very important to initialize the BSTR parameter with NULL prior to its first usage. Otherwise exception and crash. This is only true for non-inplace running servers (that is, not necessary for calls from jobdsvr.ocx, but crucial for calls of Axyz.exe). There is also no such restriction for either calls from a VB client.

Example on how to apply string returning functions

```
BSTR str = NULL; // <== very important
Axyz.GetActiveCoordSys(&str);
CString sCS(str);
```

The former low level functions AnalyzeTwoPoint(), AnalyzeAngle(), AnalyzeDistance() and AnalyzeArcDistance() were almost useless. Since these functions did not store any data, the only application was to let them print their results to Printer or Logfile. However, programmers are often interested in getting the results of these calculations into a in-core variable. The new functions with same names, but with appended ‘Ex’ do this. They act the same as the old functions, but in addition, they return their calculated (double) values through output parameters. These parameters are always the last ones in the parameter list. The other parameters are exactly the same as for the old functions. See documentation there.
One additional parameter is to point out: ‘bDoPrint’. It’s a boolean flag to indicate whether results also to be printed (according to Online Output Settings). It is important to know that this flag is ignored unless the ‘Print on Request only’ flag in the Settings is enabled.

The functions AnalyzeIntersectEx(), AnalyzePerpendEx(), AnalyzeBisectEx() and AnalyzeProjectionEx() act exactly the same as their relatives without ‘Ex’. See documentation there. The only difference is that the ‘Ex’ functions in addition return some additional output (namely some distances and angles). These values relate to the bottom output values in the related dialog boxes. See there. These are the values that are compared against global tolerance warning thresholds (Settings/Warnings)

NormalPlaneFromLine() Is a new function which has no relative at User interface level. It just creates a Plane normal to a given Line. Optionally, you can pass a point ID as third argument. That means that the created Plane will pass through this point. If the third argument is empty, then the Plane will pass through the ‘origin’ of the Line.

Please refer to the CDM TypeLibrary (Axyz.tlb) for further information

4. STM/MTM - High Level Commands

After each High Level Command the Theodolite Manager gets the focus. In order to bring script messages to the front after such a High Level Command we recommend to use the command *Me.SetFocus*.

4.1. General Commands

4.1.1. ShowTheoman

This command brings the Theodolite Manager on top of all other running tasks. It is therefore visible, even if it was hidden before.

Input Parameter	
Mode	(Boolean) Show normal (if true), Show minimized (if false)
Return Value	
Status	(Boolean)
Example	Status = TheoM.**ShowTheoman(True)** TheoM.**ShowTheoman True**

4.2. Mode Menu Commands

4.2.1. ShowBuildModeDlg

Call the “Build” Mode Dialog Box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user. Now the view is reset to Standard mode.

Input Parameter	
none	
Return Value	
Status	(Boolean)
Example	Status = TheoM.**ShowBuildMode** TheoM.**ShowBuildMode**

4.2.2. ShowInspectModeDlg

Call the “Inspect” Mode Dialog Box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user. Now the view is reset to Standard mode.

Input Parameter

none

<u>*Return Value*</u>

Status	(Boolean)
<u>*Example*</u>	Status = TheoM.**ShowInspectModeDlg**
	TheoM.ShowInspectModeDlg

4.3. View Menu Commands

4.3.1. ShowRunningAngles

Make the “View Running Data” window visible if the parameter is set to True. With the parameter set to False, the box is not visible. Also with the dialog box not visible, the latest 50 measurements are available.

<u>*Input Parameter*</u>

Status	(Boolean)

<u>*Return Value*</u>

Status	(Boolean)
<u>*Example*</u>	Status = TheoM.**ShowRunningAngles** (True)
	TheoM.**ShowRunningAngles** True

4.4. Setup Menu Commands

4.4.1. ShowStationSetup

Call the “Station Setup” dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

<u>*Return Value*</u>

Status	(Boolean)
<u>*Example*</u>	Status = TheoM.**ShowStationSetup**
	TheoM.**ShowStationSetup**

4.4.2. ShowLevelTheodolite

Call the “Level Theodolite” dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user. This command is only available for stations that have no measurements recorded.

<u>*Return Value*</u>

Status	(Boolean)

Example	Status = TheoM.**ShowLevelTheodolite**
	TheoM.**ShowLevelTheodolite**

4.4.3. ShowZeroTheodolite

Call the “Zero Theodolite” dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user. This command is only available for stations that have no measurements recorded.

Return Value

Status	(Boolean)
Example	Status = TheoM.**ShowZeroTheodolite**

4.4.4. ShowFieldCheck

Call the “Field Check” dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

Return Value

Status	(Boolean)
Example	Status = TheoM.**ShowFieldCheck**
	TheoM.**ShowFieldCheck**

4.4.5. ShowCheckTarget

Call the “Check Target” dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

Return Value

Status	(Boolean)
Example	Status = TheoM.**ShowCheckTarget**
	TheoM.**ShowCheckTarget**

4.4.6. ShowTotalStationSetup

Call the “Reflector Setup” dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

Return Value

Status	(Boolean)
Example	Status = TheoM.**ShowTotalStationSetup**
	TheoM.**ShowTotalStationSetup**

4.4.7. ShowAtmosphericSetup

Call the “Atmospheric Setup” dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

<u>*Return Value*</u>	
Status	(Boolean)
<u>*Example*</u>	Status = TheoM.**ShowAtmosphericSetup**
	TheoM.**ShowAtmosphericSetup**

4.4.8. ShowTheodoliteWarnings

Call the “Theodolite Warnings” dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

<u>*Return Value*</u>	
Status	(Boolean)
<u>*Example*</u>	Status = TheoM.**ShowTheodoliteWarnings**
	TheoM.**ShowTheodoliteWarnings**

4.4.9. ShowGeneralSettings

Call the GeneralSettings dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

<u>*Return Value*</u>	
Status	(Boolean)
<u>*Example*</u>	Status = TheoM.ShowGeneralSettings
	TheoM.ShowGeneralSettings

4.5. Orientation Menu Commands

4.5.1. ShowMeasureAccCollimation

Call the “AccurateCollimation” measurement dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

<u>*Return Value*</u>	
Status	(Boolean)
<u>*Example*</u>	Status = TheoM.**ShowMeasureAccCollimation**

TheoM.**ShowMeasureAccCollimation**

4.5.2. ShowMeasureAppCollimation

Call the “ApproximateCollimation” measurement dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

<u>*Return Value*</u>	
Status	(Boolean)
<u>*Example*</u>	Status = TheoM.**ShowMeasureAppCollimation**
	TheoM.**ShowMeasureAppCollimation**

4.5.3. ShowMeasureLocalStation

Call the “Local Orientation” measurement dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

<u>*Input Parameter*</u>	
StationList	(String)
<u>*Return Value*</u>	
Status	(Boolean)
<u>*Example*</u>	Status = TheoM.**ShowMeasureLocalStation** (“1 2”)
	TheoM.**ShowMeasureLocalStation** “1 2”

4.5.4. ShowMeasureObjectStation

Call the “Object Orientation” measurement dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

<u>*Input Parameter*</u>	
StationList	(String)
<u>*Return Value*</u>	
Status	(Boolean)
<u>*Example*</u>	Status = TheoM.**ShowMeasureObjectStation** (“1 2 3”)
	TheoM.**ShowMeasureObjectStation** “1 2 3”

4.5.5. ShowMeasureScaleBar

Call the “Scale Bar” measurement dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

Input Parameter

StationList	(String)

Return Value

Status	(Boolean)
Example	Status = TheoM.**ShowMeasureScaleBar** (“ALL”) TheoM.**ShowMeasureScaleBar** “ALL”

4.5.6. ShowMeasureHiddenPoint

Call the “Hidden Point Measurement” dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

Input Parameter

StationList	(String)

Return Value

Status	(Boolean)
Example	Status = TheoM. **ShowMeasureHiddenPoint** (“ALL”) TheoM. **ShowMeasureHiddenPoint** “ALL”

4.6. Measure Menu Commands

For all commands, which call a dialog box, the script language program flow will pause, until the dialog box is closed by the user. The normal dialog box operation can be used.

4.6.1. ShowSetAndRead

Call the “SetAndRead” dialog box.
Return Value

Status	(Boolean)
Example	Status = TheoM.**ShowSetAndRead** TheoM.**ShowSetAndRead**

4.6.2.ShowReflectorOffset

Call the “Reflector Offset” dialog box.
Return Value

Status	(Boolean)
Example	Status = TheoM.**ShowReflectorOffsetd**

TheoM. **ShowReflectorOffsetd**

4.6.3.ShowTargetThickness

Call the “Target Thickness” dialog box.

Return Value

Status	(Boolean)
Example	Status = TheoM.**ShowTargetThickness**
	TheoM. **ShowTargetThickness**

4.6.4.ShowGoLocation

Call the “Go Location” dialog box.

Return Value

Status	(Boolean)
Example	Status = TheoM.**ShowGoLocations**
	TheoM. **ShowGoLocation**

4.6.5.ShowAutoInspect

Call the “AutoInspection” dialog box.

Return Value

Status	(Boolean)
Example	Status = TheoM.**ShowAutoInspect**
	TheoM. **ShowAutoInspect**

4.7. Window Menu Commands

4.7.1. ShowOpenTheoView

Call the “New Window” dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

Return Value

Status	(Boolean)
Example	Status = TheoM.**ShowOpenTheoView**
	TheoM.**ShowOpenTheoView**

5. STM/MTM - Low Level Commands

One of the particular problems in calling low level commands, is the synchronization between the Theodolite Manager and the script language. Often it is absolutely necessary to wait until a certain command has finished before the normal flow of the scrip program can carry on.
Calls that generate an action, like **MeasureHV** will set an internal flag, that prevents all further actions to this station until the call that caused the flag to be set, has terminated.

5.1. General Commands

5.1.1. TriggerMeasurement

With the trigger flag set to **True** all the measuring commands will immediately trigger a measurement with the corresponding station. If the flag is set to **False** the script is paused, until the measurement for the selected station is executed at the station.

Input Parameter	
Trigger Flag	(Boolean)
Return Value	
Status	(Boolean)
Example	Status = TheoM.**TriggerMeasurement** (True)
	TheoM.**TriggerMeasurement** False

5.1.2. SetDisplay1

Display the supplied test string in window 1 of the selected sensors. Depending on the connected sensor type, the string is reduced in size. Since not all sensors can display all characters, the string is also treated for this.

⚠ **CAUTION:** The application does not know about the supplied strings. It is therefore possible, that shortly after the string appears, it gets overwritten by a system message.

⚠ **CAUTION:** This command does not apply toTPS1000/2000/5000 instruments..

Input Parameter	
StationList	(String)
TextToSend	(String)

Return Value	
Status	(Boolean)
Example	Status = TheoM.**SetDisplay1** ("1", "HALLO")
	TheoM.**SetDisplay1** "all", "WAIT"

5.1.3. SetDisplay2

Display the supplied test string in window 2 of the selected sensors. Depending on the connected sensor type, the string is reduced in size. Since not all sensors can display all characters, the string is also treated for this.

CAUTION: The application does not know about the supplied strings. It is therefore possible, that shortly after the string appears, it gets overwritten by a system message.

CAUTION: This command does not apply toTPS1000/2000/5000 instruments..

Input Parameter	
StationList	(String)
TextToSend	(String)
Return Value	
Status	(Boolean)
Example	Status = **TheoM.SetDisplay2** ("1", "HALLO")
	TheoM.**SetDisplay2** "all", "WAIT"

5.1.4. GetPortMap

The returned string contains for each of the connected sensors a data pair. The first number telling the port number, the second number representing the corresponding station ID, e.g. 1 12 2 13 3 18 means :

Port 1 **Error! Reference source not found.** Station 12
Port 2 **Error! Reference source not found.** Station 13
Port 3 **Error! Reference source not found.** Station 18

Return Value	
PortMap	(String)
Example	PortMap = TheoM.**GetPortMap**

5.1.5. GetNumStationsOnline

This call return the number of currently connected stations. This are all the sensors connected to a serial communication port.

Return Value

SensorsOnline (Integer)

Example SensorsOnline = TheoM.**GetNumStationsOnline**

5.1.6. RawStringCommunication

The parameter CommandString is sent to the sensor identified with the parameter StationNo. The system waits then a maximum of 5 seconds for an answer from the sensor, and returns this answer string to the caller. A trailing <CR><LF> is automatically added to the command string. Other than that no changes or checks are performed with the strings by the application. It is therefore the sole responsibility of the user to supply correct data.

Input parameter

StationNo	(Integer)
CommandString	(String)

Return Value

SensorString	(String)

Example SensorString = TheoM.**RawStringCommunication** (1, “@N13”)

See the manual “Wild Instuments online” (Document ID: G2-366-0en) for the full documentation of all available commands. This command set applies to GSI communication with 8 characters for all Wild and Leica instruments.
This manual is available via your local Leica partner.

5.2. Setup Menu Commands

5.2.1. SetHzToZero

Set the horizontal circle of the selected station’s to zero. This can only be done if there where no previous measurements recorded for the station. Otherwise, the command is not carried out, and returns with the status variable set to false.

Input Parameter

StationNumber	(Integer)	

Return Value

Status	(Boolean)

Example Status = TheoM. **SetHzToZero** (1)
TheoM. **SetHzToZero** 1

5.2.2. TotalStationSetup

Set the reflector ID for a Total Station.

Input Parameter

StationNumber	(Integer)
ReflectorId	(String)

Return Value

Status	(Boolean)

Example Status = TheoM. **TotalStationSetup** (1, “GPR1”)
TheoM.**TotalStationSetup** 1, “Tape5000”

5.2.3.SetBeepStatus

Sets the beep volume on the indicated thaodolites.

Input Parameter

StationList	(String)	
BeepLevel	(Integer)	0 = low, 1 = medium, 2 = high ???

Return Value

Status	(Boolean)

Example Status = TheoM. **SetBeepStatus** (“1 2”, 0)
TheoM. **SetBeepStatus** “ALL”, 1

5.2.4.SkipPointsBack

Skips the default point number back

Input Parameter

StationList	(String)	
NrOfPoints	(Integer)	how many points back

Return Value

Status	(Boolean)

Example Status = TheoM. SkipPointsBack(“1 2”, 3)
TheoM. SkipPointsBack “ALL”, 1

5.2.5. SkipPointsFor

Skips the default point number forward

Input Parameter		
StationList	(String)	
NrOfPoints	(Integer)	how many points forward
Return Value		
Status	(Boolean)	
Example	Status = TheoM. **SkipPointsFor**(“1 2”, 3)	
	TheoM. **SkipPointsFor** “ALL”, 1	

5.2.6. JumpToNumber

Sets the next pointer number to the indicated value

Input Parameter	
StationList	(String)
PointNr	(Integer)
Return Value	
Status	(Boolean)
Example	Status = TheoM. **JumpToNumber** (“1 2”, 3)
	TheoM. **JumpToNumber** “ALL”, 1

5.3. Orientation Menu Commands

5.3.1. MeasureAppCollimation

An approximate collimation measurement from FromStation to ToStation will be carried out. The result is stored as a collimation measurement. If the TriggerMeasurement flag is False, the script language flow will stop until the recording was carried out at the sensor. If the flag is True, the measurement is triggered from the application

Input Parameter	
FromStation	Integer)
ToStation	Integer)
Return Value	
Status	(Boolean)
Example	Status = TheoM.**MeasureAppCollimation** (1,2)
	TheoM.**MeasureAppCollimation** 1,2

i.e. measure the approximate collimation from station 1 to station 2 and store the result in the data base.

5.3.2. MeasureAccCollimation

An accurate collimation measurement including the stations FromStation and ToStation will be carried out. The results are stored as collimation measurements. The script language flow will stop until the recording has taken place. Since a valid accurate collimation measurement must be carried out in two faces, also this call must be done twice. Set the Boolean flag FirstFace to FALSE for the second measurement. If the TriggerMeasurement flag is False, the script language flow will stop until the recording was carried out at the sensor. If the flag is True, the measurement is triggered from the application

Input Parameter

From Station number	(Integer)
To Station number	(Integer)
Face one	(Boolean)

Return Value

Status	(Boolean)
Example	Status = TheoM.**MeasureAccCollimation** (1,2, TRUE)
	TheoM.**MeasureAccCollimation** 1,2, TRUE

i.e. measure the accurate collimation from station 1 to station 2 in face one and store the result in the data base.

5.3.3. MeasureDirection

Direction measurements can be carried out with this command. Depending on the “ObjectOrient” flag, they are stored as Object or Local direction measurements.
The “Direction” parameter can have one of the following values :

1	X - Direction
2	Y - Direction
3	Z - Direction

If the TriggerMeasurement flag is False, the script language flow will stop until the recording was carried out at the sensor. If the flag is True, the measurement is triggered from the application

Input Parameter

StationList	(String)
Direction	(Integer)
ObjectOrient	(Boolean)

Return Value

Status	(Boolean)
Example	Status = TheoM.**MeasureDirection** (“1 2”, 1, False)

i.e. measure the “X” direction (Local Orientation) using station 1 and 2.

TheoM.**MeasureDirection** “all”, 2, True
i.e. measure the “Y” direction (Object Orientation) with all stations.

5.3.4. MeasureToStation

Direction measurements can be carried out with this command. Depending on the “ObjectOrient” flag, they are stored as Object or Local direction measurements.
The “Distance” parameter must be supplied in the current active units. If the TriggerMeasurement flag is False, the script language flow will stop until the recording was carried out at the sensor. If the flag is True, the measurement is triggered from the application

Input Parameter

FromStationNumber	(Integer)
Distance	(Double)
ToStationNumber	(Integer)

Return Value

Status	(Boolean)
Example	Status = TheoM.**MeasureToStation** (2, 1.2, 1)
	TheoM.**MeasureToStation** 2, 2000, 1

i.e. station 2 measures in the direction of station 1 the distance between them is approximately 2000 [active length units].

5.3.5. MeasureToPoint

Direction measurements can be carried out with this command. Depending on the “ObjectOrient” flag, they are stored as Object or Local direction measurements.
The “Distance” parameter must be supplied in the current active units. If the TriggerMeasurement flag is False, the script language flow will stop until the recording was carried out at the sensor. If the flag is True, the measurement is triggered from the application

Input Parameter

FromStationNumber	(Integer)
Distance	(Double)
ToPointNumber	(Integer)

Return Value

Status	(Boolean)
Example	Status = TheoM.**MeasureToPoint** (2, 1.2, 1)
	TheoM.**MeasureToPoint** 2, 2000, 1

i.e. Station 2 measures in the direction of station 1 the distance between them is approximately 2000 [active length units].

5.3.6. MeasureScaleBar

The selected stations will measure the selected scale bar end at the given position. If the TriggerMeasurement flag is False, the script language flow will stop until the recording was carried out at the sensor. If the flag is True, the measurement is triggered from the application.

Input Parameter	
StationList	(String)
Scale Bar used	(Integer)
Scale Bar Position	(Integer)
Scale Bar Target	(Integer)
Return Value	
Status	(Boolean)
Example	Status = TheoM.**MeasureScaleBar** (“1 2”, 1, 1, 1)
	TheoM.**MeasureScaleBar** “1 2”, 1, 1, 1

i.e. measure scale bar number 1 target 1 at position 1 using station 1 and 2 and store the result in the data base.

5.4. Measure Menu Commands

5.4.1. SetPointId

Set the current point name and description for all the selected station. The information will be used when storing the next values into the data base. They will remain active until changed again. The numerical part of the point ID will be automatically incremented. The point ID is also send to the theodolite.

Input Parameter	
StationList	(String)
Point name	(String)
Description	(String)
Return Value	
Status	(Boolean)
Example	Status = TheoM.**SetPointId** (“1 2”, “workp1/ptID1”, “Description”)
	TheoM.**SetPointId** “ALL”, “wp1/tpt1”, “All is well”

5.4.2. SetMeasureParameter

Set the measurement parameter for all selected stations. The information will be used for the next measurements. They will remain active until changed by a new SetMeasureParameter call.

Input Parameter	
StationList	(String)
Bolt hole	(Boolean)
Average	(Integer)
Return Value	
Status	(Boolean)

Example Status = TheoM.**SetMeasureParameter**("1", TRUE, 2)
i.e. set station 1 for bolt hole measurement using 2 readings.
TheoM.**SetMeasureParameter** "all", FALSE, 3
i.e. set all connected station for average measurement using 3 readings.

This command applies only for Low level measurements, NOT to predefine the settings for any High level command measurements.

5.4.3. MeasureHV

All the selected stations will be recorded. If the TriggerMeasurement flag is False, the script language flow will stop until the recording was carried out at the sensor. If the flag is True, the measurement is triggered from the application. The currently active measurement parameter (**SetMeasureParameter**) will be used. The measurement is stored into the data base using the currently set point name and description (**SetPointId**). MeasureHV will measure Horizontal and Vertical angles form all selected stations. If the station is a TC instrument, no distance measurement is carried out.

Input Parameter	
StationList	(String)
Return Value	
Status	(Boolean)
Example	Status = TheoM.**MeasureHV** ("1 2 3")
	Read sensor 1 and store the result in the data base.
	TheoM.**MeasureHV** "all"

5.4.4. MeasureHVD

All the selected stations will be recorded. If the TriggerMeasurement flag is False, the script language flow will stop until the recording was carried out at the sensor. If the flag is True, the measurement is triggered from the application. The currently active measurement parameter (**SetMeasureParameter**) will be used. The measurement is stored into the data base using the currently set point name and description (**SetPointId**). MeasureHV will measure Horizontal and Vertical angles form all selected stations. If the station is a TC instrument, also a distance measurement is carried out.

Input Parameter	
StationList	(String)
Return Value	
Status	(Boolean)
Example	Status = TheoM.**MeasureHVD** ("1 2 3")
	TheoM.**MeasureHVD** "1 2 3"

i.e. read sensors 1, 2 and 3 and store the result in the data base.

5.4.5. MeasureHiddenPoint

The selected stations will measure the selected hidden point device target. The Boolean flag "IncPointID" controls the increment of the point ID. In a typical use, it is set to True for the measurement of the first target on a device. All subsequent targets on the same device are then measured with the flag set to False. Therefore preventing the system to increment the point number.
If the TriggerMeasurement flag is False, the script language flow will stop until the recording was carried out at the sensor. If the flag is True, the measurement is triggered from the application

Input Parameter	
StationList	(String)
Hidden Point used	(Integer)
Hidden Point Target	(Integer)
IncPointId	(Boolean)
Return Value	
Status	(Boolean)
Example	Status = TheoM.**MeasureHiddenPoint**("1 2", 1, 1, True)

Measure device number 1 target 1 using station 1 and 2 and store the result in the data base:

TheoM.**MeasureHiddenPoint** "1 2", 1, 1, True
TheoM.**MeasureHiddenPoint** "1 2", 1, 2, False

5.4.6. SetTryMode

Set the system into the TRY measurement mode.

Input Parameter	
FlagValue	(Boolean)
Return Value	
Status	(Boolean)
Example	Status = TheoM.**SetTryMode** (False)
	TheoM.**SetTryMode** True

This Command applies only for theodolite views, which are already open.

5.4.7. SetStandardMode

Set the system into the standard measurement mode.

Return Value	
Status	(Boolean)
Example	Status = TheoM.**SetStandardMode**
	TheoM.**SetStandardMode**

This Command applies only for theodolite views, which are already open.

5.4.8. SetContinuousMode

Set the continuous measuring mode for all sensors that share a view with the sensors specified in the StationList parameter. The allowed update time values are between 3 and 99 seconds.

CAUTION: Selecting sensors more than once (they may be in the same view as an other sensor which has been selected previously), will result in unnessecary communication.

Input Parameter	
StationList	(String)
UpdateTime	(Integer)
Return Value	
Status	(Boolean)
Example	Status = TheoM.**SetContinousMode** ("1", 3)
	TheoM.**SetContinousMode** "1", 5

This Command applies only for theodolite views, which are already open.

5.4.9. SetBuildMode

Set the system into the build mode.

Return Value

Status	(Boolean)
Example	Status = TheoM.**SetBuildMode**
	TheoM.**SetBuildMode**

This Command applies only for theodolite views, which are already open.

5.4.10. SetInspectMode

Set the system into the inspect mode.

Return Value

Status	(Boolean)
Example	Status = TheoM.**SetInspectMode**
	TheoM.**SetInspectMode**

This Command applies only for theodolite views, which are already open.

5.4.11. SetATRMode

Set the sensor(s) into the appropriate ATR mode.

Input Parameter

StationList	(String)
ATRMode	(Integer)

0	= None
1	= Disabled
2	= ATR enabled
3	= LOCK-IN enabled

Return Value

Status	(Boolean)
Example	Status = TheoM.**SetATRMode(“1 2 3”, 2)**
	TheoM. **SetATRMode “1” 2**

5.4.12. PositionXYZ

Let the appropriate sensors point to the given position x, y, z. Coordinates must be in user units and active coordinate system.

Input Parameter

StationList	(String)
X	(Double)
Y	(Double)

Z	(Double)
Return Value	
Status	(Boolean)
Example	Status = TheoM.**PositionXYZ(“1 2”, 2021.445, 1087.632, -24.816)**
	TheoM. **PositionXYZ “1” 2021.445 8009.092 29.882**

5.4.13.SetAutoPointParams

Set the automatic point positioning mode for all TDA sensors. This tells to the sensors to point to a special position after the beam has been interrupted during a delay time.

Input Parameter

AutoPointMode (Integer)

0 = disabled
1 = keep current position
2 = point to last measured point
3 = point to a measured point ID

Delay	(Integer)	{seconds}
Point ID	(String)	: ID only relevant for mode = 3
Return Value		
Status	(Boolean)	
Example	Status = TheoM.**SetAutoPointParams(3, 20, “WP1/Point5”)**	
	TheoM. **SetAutoPointParams 3 20 “WP1/Point5”**	

5.4.14.GetATRStatus

Get one sensors ATR status.

Input Parameter

Station Number (Integer)

Return Value

Status (Integer)

0 = None
1 = Disabled
2 = ATR enabled
3 = LOCK-IN enabled
4 = LOCK-IN locked

Example Status = TheoM.G**etATRStatus(2)**

5.4.15.ClearATRErrorCondition

React to an ATR error condition. If the “Retry” parameter is set to True, the system retries the action that caused the error (e.g. retries to find the reflector). If the parameter is set to False, the action is aborted and the program flow continues.

Input Parameter

Station Number	(Integer)
Retry	(Boolean)

Return Value

Status	(Boolean)

Example Status = TheoM. ClearATRErrorCondition **(True)**

5.4.16.EnableATRErrorMessageBox

If the “DoShow” parameter is set to True, the system will display a message box in case an ATR error occurs. With the parameter set to False, the only way to detect an error condition is by monitoring the “ATRErrorCondition” messages. This is the preferred setting for an automated process!

Input Parameter

DoShow	(Boolean)

Return Value

Status	(Boolean)

Example Status = TheoM. EnableATRErrorMessageBox **(True)**

5.5. Window Menu Commands

5.5.1. OpenTheoView

Open a measurement window and include the selected stations into it. If all active and oriented stations have to be included, use the parameter “ALL” instead of an station list. If any of the given stations does not currently exist in the system, it is ignored. If none of the given station’s exist, no window is opened. This is also true, if the name of the given Coordinate System is not valid. If the string for the Coordinate System is empty, the currently active system coordinate system is used. If the string for theView caption is empty, the caption is set by the Theodolite Manager to “OLE Created View”.

Input Parameter

StationList	(String)
Coordinate System Name	(String)
View Caption	(String)

Return Value

Status	(Boolean)
Example	Status = TheoM.**OpenTheoView** (“1 2 3”, “default/BASE”, “1 2 3”) TheoM.**OpenTheoView** “ALL”, “tpt15/coord1”, “All Stations”

5.5.2. CloseAllViews

Close all open measurement windows.

Return Value

Status	(Boolean)
Example	Status = TheoM.**CloseAllViews** TheoM.**CloseAllViews**

6. LTM - High Level Commands

☞ After each High Level Command the Tracker Manager gets the focus. In order to bring script messages to the front after such a High Level Command we recommend to use the command *Me.SetFocus*.

6.1. General Commands

6.1.1. ShowTrackerMan

This command brings the Tracker Manager on top of all other running tasks. It is therefore visible, even if it was hidden before.

Input Parameter	
OnTop	(Boolean)
Return Value	
Status	(Boolean)
Example	Status = TrackerM.**ShowTrackerMan (TRUE)**
	TrackerM.**ShowTrackerMan TRUE**

6.2. File and View Menu Commands

6.2.1. ShowOpenTrackerView

Call the “New Window” dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

Return Value	
Status	(Boolean)
Example	Status = TrackerM.**ShowOpenTrackerView**
	TrackerM.**ShowOpenTrackerView**

6.3. Setup Menu Commands

6.3.1. ShowStationSetup

Call the “Station” dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

Return Value

Status	(Boolean)
Example	Status = TrackerM.**ShowStationSetup**
	TrackerM.**ShowStationSetup**

6.3.2. ShowTrackerInit

Call the "Tracker Init" dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

Return Value

Status	(Boolean)
Example	Status = TrackerM.**ShowTrackerInit**
	TrackerM.**ShowTrackerInit**

6.3.3. ShowReflectorSetup

Call the "Reflectors" dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

Return Value

Status	(Boolean)
Example	Status = TrackerM.**ShowReflectorSetup**
	TrackerM.**ShowReflectorSetup**

6.3.4. ShowOffset

Call the "Reflector Offset Correction" dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

Return Value

Status	(Boolean)
Example	Status = TrackerM.**ShowOffset**
	TrackerM.**ShowOffset**

6.3.5. ShowTargetThickness

Call the "Target Thickness" dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

Return Value

Status	(Boolean)
Example	Status = TrackerM.**ShowTargetThickness**
	TrackerM.**ShowTargetThickness**

6.3.6. *ShowTrackerWarnings*

Call the “Tracker Warnings” dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

Return Value

Status	(Boolean)
Example	Status = TrackerM.**ShowTrackerWarnings**
	TrackerM.**ShowTrackerWarnings**

6.3.8. *ShowFieldCheckBallBar*

Call the “Field Check Calc/Ball Bar” dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

Return Value

Status	(Boolean)
Example	Status = TrackerM.**ShowFieldCheckBallBar**
	TrackerM.**ShowFieldCheckBallBar**

6.3.9. *ShowFieldCheckTwoFace*

Call the “Field Check Calc/Two Face” dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

Return Value

Status	(Boolean)
Example	Status = TrackerM.**ShowFieldCheckTwoFace**
	TrackerM.**ShowFieldCheckTwoFace**

6.3.10. *ShowFieldCheckADM*

Call the “Field Check Calc/ADM” dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

Return Value

Status	(Boolean)
Example	Status = TrackerM.**ShowFieldCheckADM**
	TrackerM.**ShowFieldCheckADM**

6.3.11. ShowFieldCheckTwoPoint

Call the "Field Check Calc/IFM" dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

Return Value

Status	(Boolean)
Example	Status = TrackerM.**ShowFieldCheckTwoPoint**
	TrackerM.**ShowFieldCheckTwoPoint**

6.4. Measure Menu Commands

6.4.1. ShowSetPointId

6.4.2.

Call the "Set Point Id" dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

Return Value

Status	(Boolean)
Example	Status = TrackerM.**ShowSetPointId**
	TrackerM.**ShowSetPointId**

6.4.2. ShowMeasureDefine

Call the "Measure Define" dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

Input Parameter

Measurement Method	(Integer)

The 'Measurement Method' parameter can be one of the following:

```
Continuous = 0
Grid = 1
SphereCenter = 2
CircleCenter = 3
Stationary = 4
BuildDisplay = 8
```

BuildMeasure = 9

Return Value	
Status	(Boolean)
Example	Status = TrackerM.**ShowMeasureDefine(0)**

The measuring system must not be in one of the calibration measurement modes (BallBar, TwoFace or ADM) for this command to work properly.

6.4.3. ShowGoLocation

Call the "Go Location" dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

Return Value	
Status	(Boolean)
Example	Status = TrackerM.**ShowGoLocation**
	TrackerM.**ShowGoLocation**

6.4.4. ShowLocationPoint

Call the "Location Point" dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

Return Value	
Status	(Boolean)
Example	Status = TrackerM.**ShowLocationPoint**
	TrackerM.**ShowLocationPoint**

6.4.5. ShowFindReflector

Call the "Find Reflector" dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

Return Value	
Status	(Boolean)
Example	Status = TrackerM.**ShowFindReflector**
	TrackerM.**ShowFindReflector**

6.5. Mode Menu Commands

6.5.1. ShowAutoInspectDlg

Call the “Auto Inspect” dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

Return Value	
Status	(Boolean)
Example	Status = TrackerM.**ShowAutoInspectDlg**
	TrackerM.**ShowAutoInspectDlg**

6.5.2. ShowBuildPointsDlg

Call the “Build Points” dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

Return Value	
Status	(Boolean)
Example	Status = TrackerM.**ShowBuildPointsDlg**
	TrackerM.**ShowBuildPoitnsDlg**

6.5.3. ShowBuildShapesDlg

Call the “Build Shapes” dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

Return Value	
Status	(Boolean)
Example	Status = TrackerM.**ShowBuildShapesDlg**
	TrackerM.**ShowBuildShapesDlg**

6.5.4. ShowStandardDlg

Sets the system into Standard measurement mode. There is no dialog box associated with this mode.

Return Value	
Status	(Boolean)
Example	Status = TrackerM.**ShowStandardDlg**
	TrackerM.**ShowStandardDlg**

6.5.5. ShowCADBuildDlg

Sets the system into “CAD Build” measurement mode. There is no dialog box associated with this mode.

<u>*Return Value*</u>

Status	(Boolean)
<u>*Example*</u>	Status = TrackerM.**ShowCADBuildDlg**
	TrackerM.**ShowCADBuildDlg**

6.5.6. ShowCADInspectDlg

Sets the system into “CAD Inspect” measurement mode. There is no dialog box associated with this mode.

<u>*Return Value*</u>

Status	(Boolean)
<u>*Example*</u>	Status = TrackerM.**ShowCADInspectDlg**
	TrackerM.**ShowCADInspectDlg**

6.6. Alignment Menu Commands

6.6.1. ShowEstablishDistScale

Call the “Establish Distance/Scale Bar” dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

<u>*Return Value*</u>

Status	(Boolean)
<u>*Example*</u>	Status = TrackerM.**ShowEstablishDistScale**
	TrackerM.**ShowEstablishDistScale**

6.6.2. ShowEstablishDistTwoPt

Call the “Establish Distance/Two Point” dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

<u>*Return Value*</u>

Status	(Boolean)
<u>*Example*</u>	Status = TrackerM.**ShowEstablishDistTwoPt**
	TrackerM.**ShowEstablishDistTwoPt**

6.6.3. ShowChangeFace

This function is no longer valid. Use the function “ChangeFace” instead.

6.6.4. ShowAlignTracker

Call the “Align Tracker” dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

<u>*Return Value*</u>	
Status	(Boolean)
<u>*Example*</u>	Status = TrackerM.**ShowAlignTracker**
	TrackerM.**ShowAlignTracker**

6.6.5. ShowADMInstrument

Call the “ADM/Instrument” dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

<u>*Return Value*</u>	
Status	(Boolean)
<u>*Example*</u>	Status = TrackerM.**ShowADMInstrument**
	TrackerM.**ShowADMInstrument**

6.6.6. ShowADMReflector

Call the “ADM/Reflector” dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

<u>*Return Value*</u>	
Status	(Boolean)
<u>*Example*</u>	Status = TrackerM.**ShowADMReflector**
	TrackerM.**ShowADMReflector**

6.6.7. ShowSelectAlignment

Call the “Select Alignment” dialog box. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

Return Value	
Status	(Boolean)
Example	Status = TrackerM.**ShowSelectAlignment**
	TrackerM.**ShowSelectAlignment**

7. LTM - Low Level Commands

7.1. General Commands

7.1.1. IsADMType

Allows the user to determine if the station is a type that has an ADM.

Input Parameters	
Station	(Integer)
Return Value	
Status	(Boolean)
Example	Status = TrackerM.**IsADMType(1)**

7.1.2. GetCurrentStation

Allows the user to determine what the current selected station is. This is determined by first looking at the active view for the station. If there is no measurement window open then the selected station on the station status view is returned.

Return Value	
Station	(Integer)
Example	Station = TrackerM.**GetCurrentStation**

7.1.3. GetLastErrorMsg

Retrieves the last error message received from the station specified. If one of the commands returns FALSE, then this command can be called to determine what error message caused the command to fail.

Input Parameters	
Station	(Integer)
Output Parameters	
errorNum	(Integer)
Return Value	
ErrorString	(String)
Example	ErrorString = TrackerM.**GetLastErrorMsg(1, errorNum)**

7.1.4. GetTrackerStatus

Allows the user to determine the current status of the tracker. The return is a coded integer and includes the following information.

```
00000000
||||||||- OnLine              1
|||||||-- Booted              2
||||||--- Parameters set      4
|||||---- Laser ready         8
||||----- Initialized        16
|||------ Reflector defined  32
||------- Setup OK           64
|-------- Reflector found   128
--------- Is measuring      256
```

Input Parameters	
Station	(Integer)
Return Value	
Status	(Integer)
Example	Status = TrackerM.**GetTrackerStatus(1)**

7.1.4. ChangeFace

Changes face setting of the tracker measurement. This command replaces the automation command ShowChangeFace.

Input Parameters	
Station	(Integer)
Return Value	
Status	(Boolean)
Example	Status = TrackerM.**ChangeFace(1)**
	TrackerM. **ChangeFace(1)**

7.2. File and View Menu Commands

7.2.1. CloseAllViews

Closes all open measurement windows.

Return Value	
Status	(Boolean)
Example	Status = TrackerM.**CloseAllViews**
	TrackerM.**CloseAllViews**

7.2.2. OpenTrackerView

Opens a tracker view with the parameters given without displaying the open measurement window dialog box.

Input Parameters	
StationList	(String)
Coordinate System	(String)

Window Caption	(String)

Return Value

Status	(Boolean)

Example

Status = TrackerM.**OpenTrackerView("1", "BASE","Test Title")**

7.3. Setup Menu Commands

7.3.1. InitializeTracker

Allows the user to run the initialization of the tracker via a script file.

Input Parameters

Station	(Integer)
Temperature	(Double)
Pressure	(Double)
GoHome	(Boolean)

Return Value

Status	(long)
0	no error
1	boot file not found
2	boot send error
3	boot failed
4	parameter file not found
5	invalid parameters sent
6	send parameters failed
7	self test failed
8	motors are off
9	laser is not warmed up
10	no reflector is defined
11	set reflector failed
12	setup nivel failed
13	nivel is not oriented
14	nivel is not level
15	nivel is out of tolerance
16	tracker is not ready
17	station error
18	temperature out of range
19	pressure out of range

Example Status = TrackerM**.InitializeTracker(1, 23.5, 990.6, FALSE)**

☞ Temperature and Pressure is expected to be in the current system units. If these values are out of range an error is returned.

☞ If “GoHome” is TRUE then the system will send the tracker head to point to the defined bird bath location. If this value is FALSE the initialization will stop without sending the head to a known position. The script command GoLocation can then be used to send it to any location if needed.

7.3.2. OrientToGravity

Call the “Orient to Gravity” command. The normal dialog box operation can be used. The script language program flow will pause, until the dialog box is closed by the user.

Input Parameters	
Station	(Integer)
Return Value	
Status	(Boolean)
Example	Status = TrackerM.**OrientToGravity(1)**

7.3.3. SelectReflector

Allows the user to select a reflector for the specified station. Returns FALSE if the reflector is not define, station is invalid, or the reflector doesn’t exist.

Input Parameters	
Station	(Integer)
ReflectorId	(String)
Return Value	
Status	(Boolean)
Example	Status = TrackerM.**SelectReflector(1, “cornercub1”)**

7.3.4. SetTargetThicknessParameters

Allows the user to set the individual parameters for the target thickness.

Input Parameters	
Coord System	(String)
Direction Type	(Integer)
0 – Shape	
1 – Axis	

Xvalue	(double)
Yvalue	(double)
Zvalue	(double)
Planar	(Boolean)

TRUE – Planar
FALSE – Circular

If “Direction Type” is set to Shape then the “Xvalue” parameter is the distance off the shape and the “Yvalue” and “Zvalue” parameters are ignored. The “Planar” parameter is only used if the “Direction Type” is set Shape and the coordinate system selected is a circle.

Values entered are assumed to be in the currently active units, coordinate system, and coordinate system type and will be converted appropriately before use.

Return Value

Status	(Boolean)
Example	Status = TrackerM.**SetTargetThicknessParameters (“Plane1”,1,0.5,0.0,0.0,TRUE)**

7.3.5. SetTargetThicknessActive

Allows the user to turn target thickness on or off. The SetTargetThicknessParameters must have been used prior to this command in order to insure that the parameters are correctly set.

Input Parameters

Active	(Boolean)

Return Value

Status	(Boolean)
Example	Status = TrackerM.**SetTargetThicknessActive(TRUE)** TrackerM.**SetTargetThicknessActive TRUE**

7.3.6. SetOffsetParameters

Allows the user to set the individual parameters for the reflector offset. The current reflector determines the amount of offset.

Input Parameters

Coord System	(String)
Direction Type	(Integer)

0 – Shape
1 – X Axis
2 – Y Axis
3 – Z Axis

Direction (Boolean)
TRUE – Positive direction or Shape Out
FALSE – Negative direction or Shape In

Planar (Integer)
TRUE – Planar
FALSE – Circular

☞ The “Planar” parameter is only used if the “Direction Type” is set Shape and the coordinate system selected is a circle.

Return Value

Status (Boolean)

Example

Status = TrackerM.**SetOffsetParameters(“Circle1”, 1,TRUE,TRUE)**

7.3.7. SetOffsetActive

Allows the user to turn the reflector offset on or off. The SetOffsetParameters must have been used prior to this command in order to insure that the parameters are correctly set.

Input Parameters

Active (Boolean)

Return Value

Status (Boolean)

Example Status = TrackerM.**SetOffsetActive(TRUE)**
TrackerM.**SetOffsetActive TRUE**

7.4. Measure Menu Commands

7.4.1. SetMeasureMethod

Allows the user to select the current measurement method.

Input Parameters

Measurement Method (Integer)
For valid values see ShowMeasureDefine

Return Value

Status	(Boolean)
Example	Status = TrackerM.**SetMeasureMethod(0)**
	TrackerM.**SetMeasureMethod 0**

7.4.2. GoHome

Sends the specified station to its current home position. In most cases this is the bird bath location.

The function returns the control back to Visual Basic immediately after calling. It sends out a message, as soon as the function has been finished. The message blaster can detect this message.

Input Parameters	
Station	(Integer)
Return Value	
Status	(Boolean)
Example	Status = TrackerM.**GoHome(1)**

7.4.3. GoLocationPoint

Sends the specified station to the location entered in the parameter list. The coordinates entered are assumed to be in RHR and in the current coordinate system.

The function returns the control back to Visual Basic immediately after calling. It sends out a message, as soon as the function has been finished. The message blaster can detect this message.

Input Parameters	
Station	(Integer)
Xvalue	(double)
Yvalue	(double)
Zvalue	(double)
UseADM	(Boolean)
Return Value	
Status	(Boolean)
Example	Status = TrackerM.**GoLocationPoint(1, 1.56, 2.34, 10.2, TRUE))**

7.4.4. SetPointId

Allows the user to set the next point id and descriptor for the specified station. Actions will effect the current measurement method selected.

Input Parameters	
Station	(Integer)
PointId	(String)
Description	(String)
Return Value	
Status	(Boolean)
Example	Status = TrackerM.**SetPointId**

7.4.5. SetStationaryParameters

Allows the user to set the individual parameters for a stationary measurement that are found on the measure define dialog box.

Input Parameters

Time Separation	(double)
Collection Rate	(double)
Measure Count	(Integer)
Use ADM	(Boolean)
Store All	(Boolean)

Return Value

Status	(Boolean)

Example

Status = TrackerM.**SetStationaryParameters(0.1, 0.1, 100, FALSE, FALSE)**

“Time Separation” and “Collection Rate” must be greater then 0.001. If it is not the system will use default values of 0.01 (100 pts/second).

If the
“Measure Count” is zero then the stationary point is measured until the StopMeasure command is received or the beam is broken. This is the same as if the user has selected “Continuous” on the Measure Define dialog box.

☞ If the parameter “Store All” is TRUE than all measurements collected for the single point will be stored in the database. If this parameter is FALSE then only the average of the measurements is stored into the database.

☞ This command will force the system to select the “Stationary” measurement method.

7.5. New Automation commands

7.5.1. GetVersionNumber

Returns the version number of the set of build in Automation commands. Use this check to ensure, that all the used commands are already supported.

Input Parameters
none
Return Value
VersionNumber (Integer)
Example
VersionNumber = LTM.**GetVersionNumber**

7.5.2. RestoreOLEStatus

Resets the Automation status of the running instance of the LTM. Use this call, if a previous command did not finish correctly.

Input Parameters
none
Return Value
none
Example LTM. **RestoreOLEStatus**

7.5.3. ShowErrorMessages

Disable the display of error messages during the Automation session. This flag is valid as long as the LTM is running. It is automatically reset on startup.

Input Parameters
Show (Boolean)
Return Value
none

Example LTM. **ShowErrorMessages** (False)

7.5.4. SetActiveTracker

Set the tracker with the specified network number as the active station. The trackers station number can be achieved by calling the **GetActiveStation** command.

Input Parameters

NetworkNumber	(Integer)

Return Value

Status	(Boolean)

Example Status = LTM. **SetActiveTracker (1)**

7.5.5. GetActiveStation

Returns the currently active (online) station number. This number is used as a parameter for most of the other Automation commands. Caution : this command does not return a Boolean. It returns a station number. If this number is 0 (zero) no active station was found!

Input Parameters

none

Return Value

StationNumber	(Integer)

Example CurrentStation = LTM. **GetActiveStation ()**

7.5.6. GetObjectTemperature

Returns the current object temperature as stored in the sensor module. If the Measure parameter is set to “True”, the system tries to measure the object temperature using the weather monitor program. Upon success, the sensor module value is also updated with the new value. This operation can take some time, if the weather monitor is not running! The returned object temperature value is only valid, if the status returns “True”.

Parameters

ObjectTemperature	(double)	(out)
Measure	(Boolean)	(in)

Return Value

Status	(Boolean)

Example Status = LTM. **GetObjectTemperature (ObjTempVal, True)**

7.5.7. GetAirTemperature

Returns the current air temperature as stored in the sensor module. If the Measure parameter is set to “True”, the system tries to measure the air temperature using the weather monitor program. Upon success, but only if the “UseTemperature” checkbox is checked in the “Tracker Init” dialog, the sensor module value is also updated with the new value. This operation can take some time, if the weather monitor is not running! The returned air temperature value is only valid, if the status returns “True”.

Parameters

AirTemperature	(double)	(out)
Measure	(Boolean)	(in)

Return Value

Status	(Boolean)

Example Status = LTM. **GetAirTemperature (AirTempVal, True)**

7.5.8. GetPressure

Returns the current pressure as stored in the sensor module. If the Measure parameter is set to “True”, the system tries to measure the pressure using the weather monitor program. Upon success, but only if the “UsePressure” checkbox is checked in the “Tracker Init” dialog, the sensor module value is also updated with the new value. This operation can take some time, if the weather monitor is not running! The returned pressure value is only valid, if the status returns “True”.

Parameters

Pressure	(double)	(out)
Measure	(Boolean)	(in)

Return Value

Status	(Boolean)

Example Status = LTM. **GetPressure (PressureVal, True)**

7.5.9. GetHumidity

Returns the current humidity as stored in the sensor module. If the Measure parameter is set to “True”, the system tries to measure the humidity using the weather monitor program. Upon success, the sensor module value is also updated with the new value. This operation can take some time, if the weather monitor is not running! The returned humidity value is only valid, if the status returns “True”.

Parameters

Humidity	(double)	(out)

Measure	(Boolean)	(in)

Return Value

Status	(Boolean)
Example	Status = LTM. **GetHumidity (HumidityVal, True)**

7.5.10. SetAirTemperature

Set the current air temperature as stored in the sensor module. As a consequence of this operation, the refraction indexes are calculated and the laser tracker is updated with the new values.

Input Parameters

AirTemperature	(double)

Return Value

Status	(Boolean)
Example	Status = LTM. S**etAirTemperature (AirTempVal)**

7.5.11. SetPressure

Set the current pressure as stored in the sensor module. As a consequence of this operation, the refraction indexes are calculated and the laser tracker is updated with the new values.

Input Parameters

Pressure	(double)

Return Value

Status	(Boolean)
Example	Status = LTM. S**etPressure (PressureVal)**

7.5.12. Measure

This command will eventually replace the **StartMeasure** command!

Start a measurement. The command returns immediately, use the “Message Blaster” event to synchronize the program flow of the script. The parameter **StoreResult** determines if the measurement is stored in the data base.

Input Parameters

Station	(Integer)
StoreResult	(Boolean)

Return Value

Status	(Boolean)
Example	Status = LTM. **Measure (1, False)**

LTM. **Measure** 1, True

7.5.13. Stop

This command will eventually replace the **StopMeasure** command!
Stop the currently ongoing measurement.

<u>*Input Parameters*</u>	
Station	(Integer)
<u>*Return Value*</u>	
Status	(Boolean)
<u>*Example*</u>	Status = LTM. Stop **(1)**
	LTM. **Stop**1

7.5.14. LaserOn

Switch the laser tube of the selected station On or Off. Keep in mind, switching the laser on takes about 15 to 20 minutes for the system to stabilize itself. This delay must be accounted for in the script program, because the **LaserOn** command retuns immediately.

<u>*Input Parameters*</u>	
Station	(Integer)
LaserOn	(Boolean)
<u>*Return Value*</u>	
Status	(Boolean)
<u>*Example*</u>	Status = LTM. **LaserOn (1, False)**
	LTM. **LaserOn** 1, True

7.5.15. LaserPointer

Position the selected sensor to the coordinate specified in the parameterlist of the command. Keep in mind, the system is not able to take measurements in the situation. Capture the laser beam and use the **SetIFMDistance** command or position the tracker to a target using **FindReflector** or **GoLocationPoint**.

<u>*Input Parameters*</u>	
Station	(Integer)
Xcoord	(double)
Ycoord	(double)
Zcoord	(double)
<u>*Return Value*</u>	
Status	(Boolean)

Example	Status = LTM. **LaserPointer (1, 100.3, 200,2 –120.5)**
	LTM. **LaserPointer 1, 100.3, 200,2 –120.5**

7.5.16. SetIFMDistance

Use this command to set the correct laser interferometer distance after capturing the laser beam (calling **LaserPointer**) previously.

Input Parameters	
Station	(Integer)
Return Value	
Status	(Boolean)
Example	Status = LTM. **SetIFMDistance (1)**
	LTM. **SetIFMDistance 1**

7.5.17. SetStandardMode

Put the system into "Standard measure Mode". The measurement parameters used must have been specified previously by calling the "SetStationaryParameters" command.

Input Parameters	
none	
Return Value	
Status	(Boolean)
Example	Status = LTM. **SetStandardMode ()**
	LTM. **SetStandardMode**

7.5.18. SetContinuousMode

Put the system into "Continuous measure Mode". The measurement parameters used must have been specified previously by calling one of the parameter commands.
(SetContinousParametersTime, SetContinousParametersTotalDistance or SetContinousParametersDistance)

Input Parameters	
none	
Return Value	
Status	(Boolean)
Example	Status = LTM. **SetContinousMode ()**
	LTM. **SetContinousMode**

7.5.19. SetSpherecenterMode

Put the system into “Sphere center measure Mode”. The measurement parameters used must have been specified previously by calling one of the parameter commands.
(SetSpherecenterParametersTime,
SetSpherecenterParametersTotalDistance or
SetSpherecenterParametersDistance)

Input Parameters
none
Return Value

Status	(Boolean)
Example	Status = LTM. **SetSpherecenterMode ()**
	LTM. **SetSpherecenterMode**

7.5.20. SetCirclecenterMode

Put the system into “Circle center measure Mode”. The measurement parameters used must have been specified previously by calling one of the parameter commands.
(SetCirclecenterParametersTime, SetCirclecenterParametersTotalDistance or SetCirclecenterParametersDistance)

Input Parameters
none
Return Value

Status	(Boolean)
Example	Status = LTM. **SetCirclecenterMode ()**
	LTM. **SetCirclecenterMode**

7.5.21. SetBuildPointsMode

Put the system into “Build points measure Mode”. The measurement parameters used must have been specified previously by calling the “SetStationaryParameters” command.

Input Parameters
none
Return Value

Status	(Boolean)
Example	Status = LTM. **SetBuildPointsMode ()**
	LTM. **SetBuildPointsMode**

7.5.22. SetSearchRadius

Use the specified search radius for all reflector find operations from now on. Can be changed by calling the SetSearchRadius function again. The SearchRadius value is adjusted by LTM to be between 0.001 and 0.25 m.

Input Parameters

Station	(Integer)	
SearchRadius	(double)	in meters

Return Value

Status	(Boolean)
Example	Status = LTM. **SetSearchRadius (1, 0.05)**
	LTM. **SetSearchRadius 1, 0.05**

7.5.23. SetPositionTolerance

Use the positioning tolerance value for all positioning operations from now on. Can be changed by calling the SetPositionTolerance function again.

Input Parameters

Station	(Integer)	
PositionTolerance	(double)	in meters

Return Value

Status	(Boolean)
Example	Status = LTM. **SetPositionTolerance (1, 0.0005)**
	LTM. **SetPositionTolerance 1, 0.0005**

7.5.24. SetStandardParameters

This command sets the parameter for the Standard (Stationary) measurement mode.

Input Parameters

Station	(Integer)
TimeSep	(double)
NrPoints	(Integer)
UseADM	(Boolean)

Return Value

Status	(Boolean)

Example

Status = LTM. **SetStandardParameters (1, 0.01, 100, False)**
LTM. **SetStandardParameters 1, 0.01, 100, False**

7.5.25. SetContinuousParametersTime

This command sets the parameter for the continuous measurement mode using “Time separation”. Additional parameters specify whether the data is appended to an existing set of measurements and/or if the measurement set should contain only a predetermined number of points. If the “NrPoints” parameter is set to 0 (zero) the measurement will have to be stopped by the user (StopMeasure).

<u>*Input Parameters*</u>

Station	(Integer)
TimeSep	(double)
AppendSet	(Boolean)
NrPoints	(Integer)

<u>*Return Value*</u>

Status	(Boolean)

<u>*Example*</u>

Status = LTM. **SetContinousParametersTime (1, 0.01, False, 100)**
LTM. **SetContinousParametersTime 1, 0.01, False, 100**

7.5.26. SetContinuousParametersTotalDistance

This command sets the parameter for the continuous measurement mode using “Total Distance separation”. Additional parameters specify whether the data is appended to an existing set of measurements and/or if the measurement set should contain only a predetermined number of points. If the “NrPoints” parameter is set to 0 (zero) the measurement will have to be stopped by the user (StopMeasure).

The parameter “CollectionRate” must not be smaller than 0.001 seconds, resulting in a measurement rate of 1000 points / second.

<u>*Input Parameters*</u>

Station	(Integer)
TotalDistance	(double)
AppendSet	(Boolean)
NrPoints	(Integer)
CollectionRate	(double)

<u>*Return Value*</u>

Status	(Boolean)

<u>*Example*</u>

Status = LTM. **SetContinousParametersTotalDistance (1, 0.1, False, 100, 0.01)**
LTM. **SetContinousParametersTotalDistance 1, 0.1, False, 100, 0.01**

7.5.27. SetContinuousParametersDistance

This command sets the parameter for the continuous measurement mode using “individual Distance separation”. Additional parameters specify whether the data is appended to an existing set of measurements and/or if the measurement set should contain only a predetermined number of points. If the “NrPoints” parameter is set to 0 (zero) the measurement will have to be stopped by the user (StopMeasure).

The parameter CSType must be one of the following:
RHR; LHR; SCC; SCW; CCC; CCW

The parameter “CollectionRate” must not be smaller than 0.001 seconds, resulting in a measurement rate of 1000 points / second.

Input Parameters

Station	(Integer)
CSType	(String)
XDistance	(double)
YDistance	(double)
ZDistance	(double)
AppendSet	(Boolean)
NrPoints	(Integer)
CollectionRate	(double)

Return Value

Status	(Boolean)

Example

Status = LTM. **SetContinousParametersDistance (1, “RHR”, 0.1, 1.2, -0.1,False, 100, 0.1)**
LTM. **SetContinousParametersDistance 1, “RHR”, 0.1, 1.2, -0.1,False, 100, 0.1**

7.5.28. SetSpherecenterParametersTime

This command sets the parameter for the “Spherecenter” measurement mode using “Time separation”. Additional parameters specify if the measurement should stop after a predetermined number of points. If the “NrPoints” parameter is set to 0 (zero) the measurement will have to be stopped by the user (StopMeasure).

The value parameter “FixRadius” determines whether the radius will be fixed (value > 0.0) or not.

Input Parameters

Station	(Integer)
TimeSep	(double)
FixRadius	(double)
NrPoints	(Integer)

Return Value

Status	(Boolean)

Example

Status = LTM. **SetSpherecenterParametersTime (1, 0.01, 0.0, 100)**
LTM. **SetSpherecenterParametersTime 1, 0.01, 0.0, 100**

7.5.29. SetSpherecenterParametersTotalDistance

This command sets the parameter for the “Spherecenter” measurement mode using “Total Distance separation”. Additional parameters specify if the measurement should stop after a predetermined number of points. If the “NrPoints” parameter is set to 0 (zero) the measurement will have to be stopped by the user (StopMeasure).

The value parameter “FixRadius” determines whether the radius will be fixed (value > 0.0) or not.

The parameter “CollectionRate” must not be smaller than 0.001 seconds, resulting in a measurement rate of 1000 points / second.

Input Parameters

Station	(Integer)
TotalDistance	(double)
FixRadius	(double)
NrPoints	(Integer)
CollectionRate	(double)

Return Value

Status	(Boolean)

Example

Status = LTM. **SetSpherecenterParametersTotalDistance (1, 0.1, 0.0, 100, 0.01)**
LTM. **SetSpherecenterParametersTotalDistance 1, 0.1, 0.0, 100, 0.01**

7.5.30. SetSpherecenterParametersDistance

This command sets the parameter for the “Spherecenter” measurement mode using “individual Distance separation”. Additional parameters specify if the measurement should stop after a predetermined number of points. If the “NrPoints” parameter is set to 0 (zero) the measurement will have to be stopped by the user (StopMeasure).

The value parameter “FixRadius” determines whether the radius will be fixed (value > 0.0) or not.

The parameter CSType must be one of the following:
RHR; LHR; SCC; SCW; CCC; CCW

The parameter “CollectionRate” must not be smaller than 0.001 seconds, resulting in a measurement rate of 1000 points / second.

Input Parameters

Station	(Integer)
CSType	(String)
XDistance	(double)
YDistance	(double)
ZDistance	(double)
FixRadius	(double)
NrPoints	(Integer)
CollectionRate	(double)

Return Value

Status	(Boolean)

Example

Status = LTM. **SetSpherecenterParametersDistance (1, “RHR”, 0.1, 1.2, -0.1, 0.0, 100, 0.1)**
LTM. **SetSpherecenterParametersDistance 1, “RHR”, 0.1, 1.2, -0.1, 0.0, 100, 0.1**

7.5.31. SetCirclecenterParametersTime

This command sets the parameter for the “Circlecenter” measurement mode using “Time separation”. Additional parameters specify if the measurement should stop after a predetermined number of points. If the “NrPoints” parameter is set to 0 (zero) the measurement will have to be stopped by the user (StopMeasure).

The value parameter “FixRadius” determines whether the radius will be fixed (value > 0.0) or not.

Input Parameters

Station	(Integer)
TimeSep	(double)
FixRadius	(double)
NrPoints	(Integer)

Return Value

Status	(Boolean)

Example

Status = LTM. **SetCirclecenterParametersTime (1, 0.01, 0.0, 100)**
LTM. **SetCirclecenterParametersTime 1, 0.01, 0.0, 100**

7.5.32. SetCirclecenterParametersTotalDistance

This command sets the parameter for the “Circlecenter” measurement mode using “Total Distance separation”. Additional parameters specify if the measurement should stop after a predetermined number of points. If the “NrPoints” parameter is set to 0 (zero) the measurement will have to be stopped by the user (StopMeasure).

The value parameter “FixRadius” determines whether the radius will be fixed (value > 0.0) or not.

The parameter “CollectionRate” must not be smaller than 0.001 seconds, resulting in a measurement rate of 1000 points / second.

Input Parameters

Station	(Integer)
TotalDistance	(double)
FixRadius	(double)
NrPoints	(Integer)
CollectionRate	(double)

Return Value

Status	(Boolean)

Example

Status = LTM. **SetCirclecenterParametersTotalDistance (1, 0.1, 0.0, 100, 0.01)**
LTM. **SetCirclecenterParametersTotalDistance 1, 0.1, 0.0, 100, 0.01**

7.5.33. SetCirclecenterParametersDistance

This command sets the parameter for the "Circlecenter" measurement mode using "individual Distance separation". Additional parameters specify if the measurement should stop after a predetermined number of points. If the "NrPoints" parameter is set to 0 (zero) the measurement will have to be stopped by the user (StopMeasure).

The value parameter "FixRadius" determines whether the radius will be fixed (value > 0.0) or not.

The parameter CSType must be one of the following:
RHR; LHR; SCC; SCW; CCC; CCW

The parameter "CollectionRate" must not be smaller than 0.001 seconds, resulting in a measurement rate of 1000 points / second.

Input Parameters

Station	(Integer)
CSType	(String)
XDistance	(double)
YDistance	(double)
ZDistance	(double)
FixRadius	(double)
NrPoints	(Integer)
CollectionRate	(double)

Return Value

Status	(Boolean)

Example

Status = LTM. **SetCirclecenterParametersDistance (1, "RHR", 0.1, 1.2, -0.1, 0.0, 100, 0.1)**
LTM. **SetCirclecenterParametersDistance 1, "RHR", 0.1, 1.2, -0.1, 0.0, 100, 0.1**

7.5.34. SetBuildPointData

Set the reference information for the next build measurement.

The syntax for the PointName is according to the general Axyz naming convention (e.g. "Workpiece1/Point1")

The syntax for the RefCoordSystem name is according to the general Axyz naming convention (e.g. "Default/Base"). The coordinate system **must** exist prior to this call!

Input Parameters

PointName	(String)
XRefCoordinate	(double)
YRefCoordinate	(double)
ZRefCoordinate	(double)
RefCoordSystem	(String)

Return Value

Status	(Boolean)

Example

Status = LTM.**SetBuildPointData ("WorkP/RefPt1", 100.02, 205,99, -305,9, "Default/Base")**

LTM.**SetBuildPointData "WorkP/RefPt1", 100.02, 205,99, -305,9, "Default/Base"**

7.5.35. GetLastResult

This command returns the last measured coordinate, the corresponding total RMS and maximum deviation values. The values returned are only valid, if the status returned is True. Otherwise, the values are not defined.

Input Parameters

None

Output Parameters

Xcoord	(double)	in current system units
Ycoord	(double)	and "display" coordinate
Zcoord	(double)	system transformation
TotalRms	(double)	Total RMS
MaxDev	(double)	Maximum Deviation value

Return Value

Status	(Boolean)

Example

Status = LTM. **GetLastResult (XcoordVar, YcoordVar, ZcoordVar, TotalRmsVar, MaxDevVar)**

7.5.36. SetGridMode

Put the system into "Grid Measure Mode". The measurement parameters used must have been specified previously by calling the parameter command
(SetGridParametersDistance).

Input Parameters

none

Return Value

Status	(Boolean)
Example	Status = LTM. **SetGridMode ()**
	LTM. **SetGridMode**

7.5.37. SetGridParametersDistance

This command sets the parameter for the “Grid Measurement Mode”. Additional parameters specify whether the data is appended to an existing set of measurements and/or if the measurement set should contain only a predetermined number of points. If the “NrPoints” parameter is set to 0 (zero) the measurement will have to be stopped by the user (StopMeasure).

The parameter CSType must be one of the following:
RHR; LHR; SCC; SCW; CCC; CCW

The parameter “CollectionRate” must not be smaller than 0.001 seconds, resulting in a measurement rate of 1000 points / second.

Input Parameters

Station	(Integer)
CSType	(String)
XDistance	(double)
YDistance	(double)
ZDistance	(double)
AppendSet	(Boolean)
NrPoints	(Integer)
CollectionRate	(double)

Return Value

Status	(Boolean)

Example

Status = LTM. **SetGridParametersDistance (1, “RHR”, 0.1, 1.2, -0.1,False, 100, 0.1)**

LTM. SetGridParametersDistance 1, “RHR”, 0.1, 1.2, -0.1,False, 100, 0.1

7.5.38. ContinuousTryDataTransfer

This command tells to the system to send continuous try measurements to the TcpIp line (if parameter is True) or to stop this data transfer (if False). The default system setting is False.

Input Parameters	
Enabled	(Boolean)
Return Value	
Status	(Boolean)

Example

Status = LTM.**ContinuousTryDataTransfer (True)**
LTM.ContinuousTryDataTransfer True

7.5.39. DisableAutoPoint

This command tells to the system to disable the AutoPoint functionality. The default system setting is True.

Input Parameters	
Enabled	(Boolean)
Return Value	
Status	(Boolean)

Example

Status = LTM. **DisableAutoPoint (True)**
LTM. DisableAutoPoint True

7.5.40. SwitchStation

This command allows to switch the active station from one to another. Note the difference between 'active tracker' and 'active station'. The same tracker may appear on different stations! (if you move a tracker)

This command was introduced with Service Pack #1 of Axyz 1.4.0

Input Parameters	
StationNumber	(Integer)
Return Value	
Status	(Boolean)

7.6. New LTM.INI file entry

A new LTM.INI entry (OLEDisableStartupSequence=TRUE) has been added to the [Settins] section of the ini file, to allow a fully Automation controlled startup sequence to function.

If this variable is set to FALSE, the build in startup sequence (Station editor, Reflector definition box, Tracker init box …) will pop up as usual.

8. Database Access Functions

8.1. General Database Access

8.1.1. OpenDB

This function opens an AXYZ database and seta initial data for active units, coordinate system, and coordinate system type. The function checks to see if a database is currently opened, if so it closes it and reopens a new instance.

Input Parameter

FileName	(String)

Return Value

Status	(Long)
0	No error
1	Error opening database
-1	Current CS not found

Example

Status = DB.OpenDB("C:\AXYZDATA\JOBS\LEFTWING.AXYZ")

8.1.2. CloseDB

This function closes the open AXYZ database and sets the database instance object to NULL.

Input Parameter

Return Value

Status	(Long)
0	No error

Example

Status = DB.CloseDB()

8.1.3. GetSystemSettings

This function resets data for active units, coordinate system, and coordinate system type. Call this function after every modification of such data. The function is automatically called within the functions: GetPointDataByName and GetSetDataByName. This ensures the following points be corrected and transformaed with current settings (coordinate system parameters, unit factors, reflector offsets).

Input Parameter

Return Value

Status	(Long)
0	No error
-1	Invalid Coordinate System in System Record
-100	Database not initialized

Example

Status = DB. GetSystemSettings()

8.2. Accessing Points

8.2.1. GetPointDataByName

This function reads the database for the points that match the workpiece and point ID that are input. The input strings create a WP/PtID string that is added to a CStringList. The DB Point read function from the jobdb.dll is used to read the data in a CPointList. The function accepts wild cards. The data from individual points in the list is read by calling the NextPointData function.

Input Parameter

Wp	(String)
PtID	(String)

Return Value

Status	(Long)
0	No points found
-100	Database not initialized
-102	Database Read error
> 0	Number of point in the list

Example

Status = DB.GetPointDataByName(Wp, PtID)

8.2.2. GetPointDataAfterTime

This function reads the database for the points that are later in time than the date/time input. The input strings create a WP/PtID string that is added to a CStringList. The DB Point read function from the jobdb.dll is used to read the data in a CPointList. The function accepts wild cards. The data from individual points in the list is read by calling the NextPointData function.

Input Parameter

TargetDate	(DATE)

Return Value

Status	(Long)
0	No points found

-100	Database not initialized
-102	Database Read error
> 0	Number of point in the list

Example

Status = DB.GetPointDataAfterTime(TargetDate)

8.2.3. NextPointData

This function gets the next point in the point list. It transforms the coordinate from the Base coordinate system to the current coordinate system, units and type. It also applies any target thickness or reflector offset corrections.

Input Parameter

Output Parameters

Wp	(String)	
PtID	(String)	
PtType	(Short)	
PtXYZ	(Double)	The point coordinates, transformed and with units applied (three element array)
PtStd	(Double)	The point standard deviation rotated and units applied (three element array)
Comment	(String)	The point comment
DateTime	(DATE)	The point date and time

Return Value

Status	(Long)
0	Good value
2	No records available or After last record, End of List
-1	Target thickness or reflector CS not found
-2	Reflector ID not found

Example

Status = DB.NextPointData(Wp, PtID, PtType, PtXYZ(0), PtStd(0), Comment, DateTime)

8.2.5. WritePoint

This function writes a point 's data to the database. The function will create the point if necessary.

Input Parameters

Wp	(String)	
PtID	(String)	
PtType	(String)	Following types are allowed: 2-Entered, 4-Calculated, 8-Control, 16-Hidden
PtXYZ	(Double)	The point coordinates transformed from

		Active CS to Base CS, three element array
PtStd	(Double)	The point standard deviation rotated and units applied(from Active CS to Base CS) three element array
PtDateTime	(DATA)	The point date and time
Filter to Base	(Boolean)	

If the point being input is in the Active CS, Units, and Type then set to TRUE so that transformation from active to Base CS is applied before storing. If the point being input is already in the BASE CS then set to FALSE, no transformation will be applied.

PtComment	(String)	The optional point comment
DeviceID	(Short)	The hidden device identifer number , if the point is a hidden point.

Return

0	Successful storage
1	Invalid Wp or Pt ID
2	Invalid point type
3	Missing hidden pt device number
-100	Database not initialized
-101	Database write error
-102	Database read error

Example

Status = DB.WritePoint("1$hp", "point1", 2, PtXYZ(0), PtStd(0), Now, True, "", 0)

It is invalid to add a point to the DB if the Workpiece does not currently exist. This function adds a workpiece if one does not exist for the point being added.

8.2.6. DeletePoint

This function deletes a point from the database. The function also deletes all measurements associated with the point.

Input Parameters

WP	(String)
PtID	(String)

Return Value

0	Successful deletion
1	Invalid Wp or Pt ID
-101	Database write error

Example

Status = DB.DeletePoint(Wp, PID)

Wild cards are allowed, so be careful. No warnings given.

8.2.7. ReadPointSupplemental

This function returns the target thickness and reflector offset data of a point. No wild cards are allowed.

Input Parameters

szWpId	(String)	
szPtId	(String)	

Output Parameters

szWpId	(String)	
szPtId	(String)	
nThType	(long)	Target thickness type
dThickness	(double)	Target thickness value
szTThickCSId	(String)	Target thickness coordinate system ID
dThickX	(double)	Target thickness X direction
dThickY	(double)	Target thickness Y direction
dThickZ	(double)	Target thickness Z direction
bThPlanar	(boolean)	Target thickness planar flag
szReflID	(String)	Reflector offset reflector ID
szReflCSId	(String)	Reflector offset coordinate system ID
nReflOffsetType	(long)	Reflector offset type
bReflPlanar	(boolean)	Reflector offset planar flag

Return Value

0	Successful read
1	Invalid Wp or Pt ID
2	(not used)
3	Target thickness coordinate system not found
4	Reflector not found
5	Reflector offset coordinate system not found
-100	Database not initialized
-102	Database read error (point not found)

Example

Status = DB.ReadPointSupplemental(szWpId, szPtId, nThType, dThickness, szTThickCSId, dThickX, dThickY, dThickZ, bThPlanar, szReflID, szReflCSId, nReflOffsetType, bReflPlanar)

8.2.8. WritePointSupplemental

This function writes the target thickness and reflector offset data to a point. No wild cards are allowed.

Input Parameters

szWpId	(String)	
szPtId	(String)	
nThType	(long)	Target thickness type
dThickness	(double)	Target thickness value
szTThickCSId	(String)	Target thickness coordinate system ID
dThickX	(double)	Target thickness X direction
dThickY	(double)	Target thickness Y direction
dThickZ	(double)	Target thickness Z direction
bThPlanar	(boolean)	Target thickness planar flag
szReflID	(String)	Reflector offset reflector ID
szReflCSId	(String)	Reflector offset coordinate system ID
nReflOffsetType	(long)	Reflector offset type
bReflPlanar	(boolean)	Reflector offset planar flag

Return Value

0	Successful write
1	Invalid Wp or Pt ID
3	Target thickness coordinate system not found (Successful write!)
4	Reflector not found
5	Reflector offset coordinate system not found (Successful write!)
6	Invalid target thickness coordinate system ID
7	Invalid reflector offset coordinate system ID
8	Invalid reflector ID
-100	Database not initialized
-101	Database write error

Example

Status = DB.WritePointSupplemental(szWpId, szPtId, nThType, dThickness, szTThickCSId, dThickX, dThickY, dThickZ, bThPlanar, szReflID, szReflCSId, nReflOffsetType, bReflPlanar)

8.2.9. NextPointDataEx

This function is an extension of the function NextPointData. It returns more parameters than the simple one.

Input Parameter

Output Parameters

Wp	(String)	
PtID	(String)	
PtType	(Short)	
PtXYZ	(Double)	The point coordinates, transformed and with units applied (three element array)

PtStd	(Double)	The point standard deviation rotated and units applied (three element array)
RMSXYZ	(Double)	The RMS values (three element array)
TotalRMS	(Double)	The total RMS value
AppexAngle	(Double)	The appex angle value in current unit
MeanError	(Double)	The mean error value (dimensionless)
PointErrDis	(Double)	The pointing error distance component (current unit)
PointErrAng	(Double)	The pointing error angle component (current unit)
Comment	(String)	The point comment
DateTime	(DATE)	The point date and time

Return Value

Status	(Long)
0	Good value
2	No records available or After last record, End of List
-1	Target thickness or reflector CS not found
-2	Reflector ID not found

Example

Status = DB.NextPointData(Wp, PtID, PtType, PtXYZ(0), PtStd(0), Comment, DateTime)

8.3. Accessing Reference File Points

8.3.1. GetRefPointDataByName

This function builds data in a CRefPointList for the points that match the referenceId, reference workpiece and point ID that is input. The input strings must match SQL conventions, that is, the wild card characters * and ? are valid for the workpiece and point ID. For example, "co*" gets all points that start with "co". Wild cards for the reference filename workpiece are not supported. The data from individual reference points is read by calling the NextRefPointData function.

Input Parameters

FileRefID	(String)	Reference Filename workpiece (cannot be a wildcard)
PtWP	(String)	
PtID	(String)	

Return Value

Status	(Long)
0	No points found
> 0	number of point in the list

-100	Database not initialized
-102	Database read error

Example

Status = DB.GetRefPointDataByName(FileRefID, Wp, PID)

8.3.2. NextRefPointData

This function gets the next reference point in the list.
You should no longer use this function. It is still available for backward compatibility.
Attention: This function has been replaced by *NextRefPointDataEx*.

Output Parameters

RefFileID	(String)	The reference file workpiece name
WPID	(String)	
PtId	(String)	
XYZ	(Double)	The reference point coordinates as stored in database with units applied, received in a 3 element array
Std	(Double)	The reference point standard deviation with units applied, received in a 3 element array
Comment	(String)	The reference point comment

Return Value

Status	(Long)
0	Good value
1	No records available or End of List

Example

Status = DB.NextRefPointData(RefFileID, WPID, PtId, XYZ(0), Std(0), Comment)

8.3.2. NextRefPointDataEx

This function gets the next reference point in the list.

Output Parameters

RefFileID	(String)	The reference file workpiece name
WPID	(String)	
PtId	(String)	
XYZ	(Double)	The reference point coordinates as stored in database with units applied, received in a 3 element array
Std	(Double)	The reference point standard deviation with units applied, received in a 3 element array

StdevFlags	(Short)	StDev Flags for X, Y and Z
Comment	(String)	The reference point comment

<u>*Return Value*</u>

Status	(Long)
1	Good value
1	No records available or End of List

<u>*Example*</u>

Status = DB.NextRefPointDataEx(RefFileID, WPID, PtId, XYZ(0), Std(0), flag(0), Comment)

8.3.3. WriteReferencePoint

This function writes a single reference point to the database. It also applies the conversion from active units to base units.
You should no longer use this function. It is still available for backward compatibility.

Attention: This function has been replaced by *WriteReferencePoinEx.*

<u>*Input Parameters*</u>

RefFileId	(String)	Reference file identifier
WpId	(String)	
PtId	(String)	
XYZ	(Double)	X, Y, and Z component of the reference point
Std	(Double)	Standard deviations for X, Y, and Z components
Comment	(String)	Reference Point descriptor

<u>*Return Value*</u>

Status	(Long)
0	Success
-100	Database not initialized
-102	Database write error

<u>*Example*</u>

Status = DB.WriteReferencePoint(“fileid”, “DEFAULT”, “point1”, XYZ(0), Std(0), “”)

8.3.3. WriteReferencePointEx

This function writes a single reference point to the database. It also applies the conversion from active units to base units. In addition to the previous function StdevFlags are passed (0 = NotFixed, 1 = Fixed, 2= Weighted).
Attention: If the flag is set to Fixed or NotFixed its Std must be zero.

Input Parameters

RefFileId	(String)	Reference file identifier
WpId	(String)	
PtId	(String)	
XYZ	(Double)	X, Y, and Z component of the reference point
Std	(Double)	Standard deviations for X, Y, and Z Components
StdevFlags	(Short)	StDev Flags for X, Y and Z
Comment	(String)	Reference Point descriptor

Return Value

Status	(Long)
0	Success
-100	Database not initialized
-102	Database write error
-103	illegal nonzero Std

Example

Status = DB.WriteReferencePointEx(“fileid”, “DEFAULT”, “point1”, XYZ(0), Std(0), flags(0), “”)

8.3.4.NextRefPointDataIJK

This function gets the next reference point in the list. It returns the normal vector components of the reference point.

Output Parameters

RefFileID	(String)	The reference file workpiece name
WPID	(String)	
PtId	(String)	
XYZ	(Double)	The reference point coordinates as stored in database with units applied, received in a 3 element array
IJK	(Double)	The reference point normal vector components, received in a 3 element array
Comment	(String)	The reference point comment

Return Value

Status	(Long)	
2		No records available or End of List
0		Good value

Example

Status = DB.NextRefPointDataIJK(RefFileID, WPID, PtId, XYZ(0), IJK(0), Comment)

8.3.5. DeleteRefPoint

This function deletes a reference point from the database.

Input Parameters

RefID	(String)
RefWPID	(String)
PtID	(String)

Return Value

0	Successful deletion
-100	Database not intialized
-101	Database write error

Example

Status = DB.DeleteRefPoint(RefID, RefWpID, PtID)

Wild cards are allowed, so be careful. No warnings given.

8.4. Accessing Hidden point Devices

8.4.1. ReadLinearDevice

This function reads a linear hidden point device from the database.

Input Parameters

RodId	(Integer)	Hidden point Rod number

Output Parameters

NoPoints	(Integer)	Number of points on the hidden point rod
RodOffset	(Double)	Radius of sphere in case of a ball like tip point
ExCoeff	(Double)	Expansion Coefficient of the rod material
StdTemp	(Double)	Rod standard calibration temperature
ActTemp	(Double)	Rod temperature during measurement
RodPoints	(Double)	Device Points with distances
RodSds	(Double)	Device Points distances Sds
applyUnits	(Boolean)	Apply temperature and length units when reading

Return Value

Status	(Long)
0	Successful
1	Device not found
-102	Database read error

Example

Status = DB.ReadLinearDevice(1, NoPoints, RodOffset, ExCoeff, StdTemp, ActTemp, RodPoints(0), RodSds(0), applyUnits)

8.4.2. CalcLinearHiddenPoint

This function calculates a linear hidden point and stores the resulting hidden point in the database.

Input Parameters

Wp1	(String)	
Id1	(String)	
Wp2	(String)	
Id2	(String)	
DeviceId	(Integer)	The device id of the hidden rod

Return Value

Status	(Long)
0	Successful storage
1	Invalid Wp or Pt ID
2	Device point not found
3	Missing hidden pt device number
4	error reading hidden pt device info
5	error in calculating hidden point mathematics
-100	Database not initialized
-102	Database write error

Example

Status = DB.CalcLinearHiddenPoint(“1$Default”, “point1”, “2$Default”, “point1”, 1)

8.4.3. WriteLinearDevice

This function writes a linear hidden point device to the database. This function also creates and stores the device point in the OffsetRodDefinition table for the number of rod points.

Input Parameters

HidRodId	(Integer)	Hidden point Rod number
NoPoints	(Integer)	Number of points on the hidden point rod
RodOffset	(Double)	Radius of sphere in case of a ball like tip point
ExCoeff	(Double)	Expansion Coefficient of the rod material
StdTemp	(Double)	Rod standard calibration temperature
ActTemp	(Double)	Rod temperature during measurement
RodPoints	(Double)	Device Points distances
RodPointSds	(Double)	Distance Standard deviations
Comment	(String)	Rod descriptor

Return Value

Status	(Long)

0	Successful
-101	Database write error

Example

Status = DB.WriteLinearDevice(1, 2, RodOffset, ExCoeff, StdTemp, ActTemp, RodPoints(0), RodPointSds(0), “”)

8.5. Accessing Coordinate Systems

8.5.1. CoordinateSystem

This function returns the coordinate active system name in the Axyz database.

Return Value

C.S.	(String)	The name including Workpiece and CS ID with a slash between them. Example: Default/BASE

Example

sCS = DB.CoordinateSystem()

8.5.2. ChangeCoordinateSystem

This function changes the current active coordinate system name in the Axyz database. It also updates the global translations, rotations, and scale factor for the active CS.

Input Parameter

New c.s. name	(String)	The name will be the Workpiece and CS ID with a slash between them. Example: Default/BASE

Return Value

Status	(Long)
0	Successful
-1	Coordinate System not found
-100	Database not initialized
-101	Database write error

Example

Status = DB.ChangeCoordinateSystem(“Default/BASE”)

8.5.3. GetCoordSysByName

This function reads the database and builds data in a CShapeList for the coordinate systems that match the workpiece and coordinate system ID that are input. The input strings must match SQL conventions. Names with wild card characters * and ? are valid. For example, "co*" gets all that start with "co". The data for individual coordinate systems is read by calling the NextCSData function.

Input Parameter

WP	(String)
CsID	(String)

Return Value

Status	(Long)
0	CS found
-1	CS not found
-100	Database not initialized
-102	Database read error

Example

Status = DB.GetCoordinateSystemByName(Wp, CsID)

8.5.4. NextCSData

This function gets the data for next coordinate system in the list.

Output Parameters

Wp	(String)	
Id	(String)	
ShType	(Integer)	Type of shape or transformation
Rotation	(Double)	Rotations
Translation	(Double)	Translations
Scale	(Double)	Scale
SizeParam	(Double)	Size parameters
DateTime	(Date)	The Cs date and time
RmsError	(Double)	Total Rms error in active units
Comment	(String)	The CS description

Return Value

Status	(Long)
0	Good Value
2	No records available or after last record, End of List

Example

Status = DB.NextCSData(Wp, Id, ShType, Rotation(0), Translation(0), Scale, SizeParam, DateTime, RmsError, Comment)

8.5.5. WriteCoordinateSystem

This function writes a coordinate system to the database.

Input Parameters

Wp	(String)	
Id	(String)	

Type	(Integer)	Type of shape or transformation
		Valid types include:
		Line = 2
		Plane = 4
		Cylinder = 16
		Paraboloid = 32
		Sphere = 64
		Cone = 128
		Scale = 256
		Translation = 512
		Rotation = 1024
		Bestfit = 2048
		Alignment = 4096

Rotate	(Double)	The rotations of the CS, 3 element array
Translate	(Double)	The translations of the CS, 3 element
Sfactor	(Double)	Scale factor of CS
SizeParam	(Double)	Size parameter for shapes
DateTime	(Date)	The date and time of the shape
Comment	(String)	Any comment to be associated with the CS

Return Value

Status	(Long)
0	Successful storage
2	Invalid shape type
-100	Database not initialized
-101	Database read error

Example

Status = DB.WriteCoordinateSystem(Wp, Id, 2048, Rotate(0), Translate(0), 1#, 1#, Now, RmsError, “”)

8.5.6. DeleteCoordSystem

This function deletes a coordinate system from the database.

Input Parameters

WP	(String)
ID	(String)

Return Value

Status	(Long)
0	Successful deletion

-100	Database not initialized
-101	Database write error

Example

Status = DB.DeleteCoordSystem(“Default”, “BASE”)

8.6. Accessing Units Information

8.6.1. LengthUnits, AngleUnits

These functions return the current active length or angle units name from the Axyz database.

Return Value

Units name	(String)	Name of length or angle units. Examples: meter, inch, feet, yard, radian, degree, gon

Examples

sLenName = DB.LengthUnits()
sAngName = DB.AngleUnits()

8.6.2. ChangeLengthUnits, ChangeAngleUnits

These functions change the current active length or angle units names in the Axyz database.

Input Parameters

New units name	(String)	The angle or length unit name will be an existing units identifier as found in the Units Table. If the user has created user defined units, those are also available. The default existing units names are:

Length:	feet	Angle:	degree
	inch		gon
	meter		radian
	micron		
	millimeter		
	yard		

Return Value

Status	(Long)
0	Successful
-1	Unit name not found
-100	Database object not initialized
-101	Database write error

Examples
Status = DB.ChangeLengthUnits(“inch”)
Status = DB.ChangeAngleUnits(“degree”)

8.6.3. LengthDigits, AngleDigits

These functions return the current active length or angle displayed decimal digits from the Axyz database.

Return Value

Digits	(Integer)	The number of digits displayed

Examples
iLenDigits = DB.LengthDigits()
iAngDigits = DB.AngleDigits()

8.6.4. SetAngleDigits, SetLengthUnits

These functions change the current active length or angle displayed decimal digits in the Axyz database.

Input Parameters

Digits	(Integer)	The number of digits to display after the decimal point. The valid range is 0-9.

Examples
bStatus = DB.SetLengthDigits(3)
bStatus = DB.SetAngleDigits(2)

8.6.5. AngleConversion, LengthConversion

These functions return the current active length or angle units conversion factor from the Axyz database.

Return Value

Factor	(Double)	Current length or angle units conversion factor

Examples
dLenFactor = DB.LengthConversion()
dAngFactor = DB.AngleConversion()

8.6.6. AngleUnitsSymbol, LengthUnitsSymbol

These functions return the current active length or angle units symbol from the Axyz database.

Return Value

Symbol	(String)	Current length or angle units symbol

Examples

sLenSymbol = DB.LengthUnitsSymbol()
sAngSymbol = DB.AngleUnitsSymbol()

8.7 Accessing Set Data

Use the functions GetSetNames and NextSetName to find out the names of the set. After that call GetSetDataByName with a distinct set name and finally read the set points using NextSetPointData within a loop.

These functions all work for “Continuous measurement sets” as well as for “Coordinate sets”. The first type is a continuous set measured with Laser Tracker. The second type is a set of points which has been calculated or manually entered or imported but which is not measured.

Note that in the following descriptions HVD means the tripel of 3 double values Horizontal direction, Vertical angle and Distance. XYZ means the coordinate tripel of any coordinate system type.

8.7.1.GetSetNames

This function builds a list of set names found in the database. The names from this list are read by calling the NextSetName function. The function reads both continuous measurement and coordinate sets.

Input Parameters

szWpID	(String)	Workpiece ID (wild card admitted)
szSetID	(String)	Set ID (wild card admitted)

Return Value

Status	(Long)
0	No sets found
> 0	number of set names in the list
-100	Database not initialized
-102	Database read error

Example

Status = DB.GetSetNames(szWpID, szSetID)

8.7.2.NextSetName

This function returns the next set name in the set name list.

Output Parameters

szWpID	(String)	Workpiece ID
szSetID	(String)	Set ID

Return Value

Status	(Long)
0	Good value
2	No records available or After last record, End of List
-100	Database not initialized

Example

Status = DB.NextName(szWpId, szSetId)

8.7.3.GetSetDataByName

This function builds data in a set point data list for the points of ***one*** set. Wild cards are not admitted. The data from individual set points are read by calling the NextSetPointData function. If no points are found but the set is measured, verify that the station is oriented. Otherwise points can not be calculated. The function focuses on points but does not deliver measurements with HVD only.

Input Parameters

szWpID	(String)	Workpiece ID (NO wild card)
szSetID	(String)	Set ID (NO wild card)

Return Value

Status	(Long)
0	No points found
> 0	number of point in the list
-100	Database not initialized
-102	Database read error

Example

Status = DB.GetSetDataByName(szWpID, szSetID)

8.7.4.NextSetPointData

This function returns the next set point in the set point data list. It transforms the coordinate from the Base coordinate system to the current coordinate system, units and type. It also applies any target thickness or reflector offset corrections. Note that HVD are only valid if the bMeas flag is true. HVD only are valid if the set is measured (continuous measurement set). Coordinate sets have no HVD.

Input Parameter

Output Parameters

szWpID	(String)	Workpiece ID

szSetID	(String)	Set ID
nIDNr	(long)	Numeric identifier of point within set
dCoord	(double array 3)	XYZ
dStdDev	(double array 3)	standard deviations XYZ
dMeas	(double array 3)	HVD
bMeas	(boolean)	HVD valid if True
bUseFlg	(boolean)	Use flag

Return Value

Status	(Long)
0	Good value
2	No records available or After last record, End of List
-100	Database not initialized

Example

Status = DB.NextSetPointData(szWpId, szSetId, nIDNr, dCoord, dStdDev, dMeas, bMeas, bUseFlg)

8.7.5.UseSetPointData

This function sets or resets the "Use" flag of a set point (continuous measurement set or coordinate set). This point must not be prepared with the function GetSetDataByName. The function accesses the database directly. Nevertheless you must know the nIDNr of the point in the set.

Input Parameter

szWpID	(String)	Workpiece ID
szSetID	(String)	Set ID
nIDNr	(long)	Numeric identifier of point within set
bUse	(boolean)	Use flag to be set

Return Value

Status	(Long)
0	Success
1	Invalid Wp or Set ID
2	Point not found
-100	Database not initialized
-101	Database write error

Example

Status = DB. UseSetPointData (szWpId, szSetId, nIDNr, bUse)

8.8 Accessing Reflector Data

8.8.1 GetReflectors

This function builds a list of all reflectors found in the database.

Input Parameters

None

Return Value

Status	(Long)
0	No sets found
> 0	number of set names in the list
-100	Database not initialized
-102	Database read error

Example

Status = DB.GetReflectors()

8.8.2 NextReflectorData

This function gets the next reflector data in the list. Do not use another data access function like GetPointDataByName and NextPointData during a loop over the NextReflectorData. Otherwise the position in the reflector data loop goes lost.

Output Parameters

ReflectorID	(String)	Name of the reflector
Type	(Integer)	Reflector type, see list below
Description	(String)	Reflector Description
Offset	(Double)	The Offset value of the reflector
DistOffset	(Double)	The distance Offset value of the reflector

Return Value

Status	(Long)
2	No records available or End of List
0	Good value

Example

```
Status = DB.NextReflectorData(szReflId, nType, szDescr,
                                         dOffset, dDistOffs)
```

8.9 Accessing Other Data

8.9.1 GetComparisonTolerances

This function gets the point comparison tolerances from the job setup (see Dialogbox 'Settings/AnalysisWarnings' in CDM).

Input Parameters

None

Output Parameters

dTolXYZ	(double array 3)	Tolerances XYZ
dTotalTol	(double)	Total Tolerance

Return Value

Status	(Long)
0	success
-100	Database not initialized
-102	Database read error

Example

Status = DB.GetComparisonTolerances(dTolXYZ, dTotalTol)

8.9.2 DeleteWorkpiece

This function deletes a workpiece with all its data: points, measurements, shapes, sets. Note that the active workpiece cannot be deleted. The return value is “DB Write Error” in this case.

Input Parameters

szWpID	(String)	Workpiece ID (wild card NOT admitted)

Output Parameters

None

Return Value

Status	(Long)
0	success
1	Invalid szWpID (empty or wild card)
-100	Database not initialized
-101	Database write error

Example

Status = DB.DeleteWorkpiece(“Wp1”)

8.9.3 GetStationOriStatus

This function returns the orientation status of a station. Note that the current Axyz system has numeric station ID, but treated as string.

Input Parameters

szStID	(String)	Station ID (wild card NOT admitted)

Output Parameters

nOriStatus	(short)	Orientation status

a.	no Measurements
b.	measurements but not oriented
c.	oriented
d.	station can be oriented
e.	station is oriented and scaled
f.	stations is oriented in control system
g.	station is oriented but not scaled
h.	station is oriented by user entered parameters1

Return Value

Status	(Long)
0	Success
1	Invalid station ID (empty or wild card)
-100	Database not initialized
-102	Database read error

Example

Status = DB. GetStationOriStatus ("1", nOriStatus)

8.9.4 GetJobInfo

This function returns the ID and the title of the currently open job.

Input Parameters

None

Output Parameters

szJobID	(string)	Job ID (name)
szJobTitle	(string)	Job title

Return Value

Status	(Long)
0	Success
-100	Database not initialized
-102	Database read error

Example

Status = DB.GetJobInfo(szJobID, szJobTitle)

8.9.5 GetJobInfoEx

This function returns the ID and the title of the currently open job as the function GetJobInfo does. In addition, the 'Job Description' is provided.

This command was introduced with **Service Pack #1** of Axyz 1.4.0

Input Parameters

None

Output Parameters

szJobID	(string)	Job ID (name)
szJobTitle	(string)	Job title
szJobDescription	(string)	Job description

Return Value

Status	(Long)
0	Success
-100	Database not initialized
-102	Database read error

Example

Status = DB.GetJobInfo(szJobID, szJobTitle, szJobDescript)

9. Demo Scripts

9.1. General remarks

This section describes simple examples to run an Orientation measurement. The examples *Guided Accurate Collimation Orientation* (High level and low level) below are quite simple.

You will find other examples in the *Scripts* subdirectory of the ***Axyz*** data directory.

Each module has an ‘Autoexec’ capability to start a script automatically after the module startup or initialization has been finished. The module will look for the exe-files as mentioned below and automatically start these scripts:

- *Autoexec.exe* for the Core data module CDM
- *AutoTM.exe* for the theodolite manager STM / MTM
- *AutoLTM.exe* for the Laser Tracker module.

Version 1.2.0 supplies improved scripts, some of the are even multi lingual:

9.1.1. General scripts (Icon Axyz.ico)

- CreateMasterDatabase(MultiLanguage).exe: Guides you through several dialog boxes in order to set-up individual Master databases for any application.
- Also attached is a project Script Template.vbp which you may consider as kind of template file to create your own applications. It contains the most typical functions and declarations and is the basis for all STM/MTM and LTM scripts.

9.1.2. CDM related scripts (Icon W95mbx04.ico)

- CDMSample(LowLevel).exe: Demonstrates automatic transformation and analysis functions without any user interrupt (except error messages). Use this script on an ‘Untitled’ job to generate sample data for the script CDMSample(HighLevel).exe.
- CDMSample(HighLevel).exe: Demonstrates transformation and analysis functions with the possibility of user interaction, i.e. it brings up one dialog after another and pre-sets the dialog options.

9.1.3. STM/MTM related scripts (Icon Theoman.ico)

- Axis Alignment.exe: Guides you step by step through three possible types of an axis alignment (Horizontal plane, standard 3-2-1 and defining a new system with a point, an axis and a plane. For more information a context-sensitive help file is implemented.
- OpenViewForECDSUsers.exe: Opens a new MTM-view in continuous mode. This script corresponds to the well-known function key F5 for ECDS users.
- Orientation(MultiLanguage).exe: Shows all necessary orientation functions for MTM on one form and gives basic status checks on the orientation procedure.

9.1.4. LTM related scripts (Icon LTD500.ico)

- ADM Calibration.exe: Not yet thoroughly tested.
- Align Tracker.exe: Not yet thoroughly tested
- Two Point Distance.exe: Not yet thoroughly tested

9.2. Guided Accurate Collimation Orientation (High Level)

The following script relates to the object FORM1 on the event LOAD.

```
' Initialize Script
STMMTM = True
LTM = True
Call ConnectAxyz(STMMTM, LTM, “English”)
Call Initialize(True, Form1)

' Demo script starts here
Form1.SetFocus
MsgBox "This Demonstration shows an Accurate Collimation_
   Measurement using the Built-in Dialogue Boxes",_
   vbOKOnly, "MTM/STM Script Language Demonstration"

' Open the running angles window
TheoM.ShowRunningAngles True

' Call the Accurate Collimation Box
TheoM.ShowMeasureAccCollimation

' Call the Scale Bar Box for all connected station's
TheoM.ShowMeasureScaleBar "ALL"

' Set the Point Id for station 1 and 2
TheoM.SetPointId "1 2", "default/cp1", "Common point"

' Call the SetAndRead Box to do the measurements
TheoM.ShowSetAndRead

' Call the Bundle Program and implement the actual path
```

```
CoreM.CallTask "c:\Program Files\Leica\Axyz\bundle.exe"

' Close the Running Angles Window
TheoM.ShowRunningAngles False
End
```

9.3. Guided Accurate Collimation Orientation (Low Level)

The following script relates to the object FORM1 on the event LOAD.

```
' Initialize Script
STMMTM = True
LTM = True
Call ConnectAxyz(STMMTM, LTM, “English”)
Call Initialize(True, Form1)

' Demo script starts here
Form1.Setfocus
MsgBox "Accurate Collimation Measurement using Low Level_
   Commands", vbOKOnly, "MTM/STM Script Language_
   Demonstration"

' Open the running angles window
TheoM.ShowRunningAngles True

' Trigger the measurements from the application
TheoM.TriggerMeasurement True

' Measure Collimation
' Alert user for first collimation measurement
Form1.SetFocus
MsgBox "Aim Stations for Forward Measurement, then press_
   OK", vbInformation, "Measure Accurate Collimation"_
   TheoM.MeasureAccCollimation 1, 2, True

' Alert user for second collimation measurement
Form1.SetFocus
MsgBox "Aim Stations for opposite Measurements, then press_
   OK", vbInformation, "Measure Accurate Collimation"_
   TheoM.MeasureAccCollimation 1, 2, False

' Measure Scale Bar
' Alert user for measurement to first end
Form1.SetFocus
MsgBox "Aim the Sensors at End 1, then press OK",_
   vbInformation, "Measure Scale Bar 1"
TheoM.MeasureScaleBar "1 2", 1, 1, 1

' Alert user for measurement to second end
Form1.SetFocus
MsgBox "Aim the Sensors at End 2, then press OK",_
   vbInformation, "Measure Scale Bar 1"
TheoM.MeasureScaleBar "1 2", 1, 2, 1

' Measure Common Points
' The measurements will be taken in a loop
For counter = 1 To 5
   WpPtId = "wpt102/autest" +Format(counter)
   Form1.SetFocus
   Msg = "Do you want to measure point : " + WpPtId + " ?"
   Status = MsgBox(Msg, vbQuestion  + vbYesNo, "Measure_
      Common Point")
   If (Status = vbYes) Then
   ' Alert user for the measurement
   Form1.SetFocus
   Msg = "Aim the Sensors at Point : " + WpPtId + " then_
      press OK"
   MsgBox Msg, vbInformation, "Measure Common Point"
      TheoM.SetPointId "all", WpPtId, "Measurement from_
          Application"
      TheoM.MeasureHV "ALL"
   End If
Next

' Call the Bundle Program and implement the actual path
' for BUNDLE.EXE
CoreM.CallTask "c:\Program Files\Leica\Axyz\bundle.exe"

' Close the running angle window
TheoM.ShowRunningAngles False

End
```