Minitab® 18

Minitab Automation

Minitab

Release 18.1.0

Contents

1 Getting Started

Introducing Automation in Minitab

The COM automation library contains a set of standard COM (Component Object Model) objects that expose much of Minitab's internal functionality. You can use this COM library with any COM-compliant language.

Strategies for Handling Errors in COM overview

HRESULT values

COM returns an HRESULT value for all methods in all component interfaces. An HRESULT indicates whether a COM method succeeded or failed. HRESULTs also report any errors in making function calls or interface method calls and identify the facilities associated with the errors, such as RPC, WIN32, or ITF for interface-specific errors. Lastly, system APIs provide a lookup from an HRESULT to a string that describes the error condition.

Using methods that return HRESULTs is fundamental to well-written components and is essential to the debugging process. Microsoft Visual Basic automatically defines each method with an HRESULT as a return. In Microsoft Visual C++, you must explicitly return an HRESULT.

ErrorInfo objects

ErrorInfo objects are often called COM exceptions because they allow an object to pass (or throw) rich error information to its caller, even across apartment boundaries. The value of this generic error object is that it supplements an HRESULT, extending the type of error description, the source of the error, and the interface identifier of the method that originated the error. You can also include pointers to an entry in a Help file.

Automation provides three interfaces to manage the error object:

- Components must implement the ISupportErrorInfo interface to advertise their support for the ErrorInfo object.
- When an error occurs, the component uses the ICreateErrorInfo interface to initialize an error object.
- After the caller inspects the HRESULT and finds that the method call failed, it queries the object to see whether it supports the ErrorInfo object. If it does, the caller uses the IErrorInfo interface to retrieve the error information.

Visual Basic programmers have easy access to the ErrorInfo object, which is exposed through the Err object. You can raise errors with the Err Raise function and catch errors with the On Error statement. The Visual Basic run-time layer takes care of the mapping for you. If you are using the Visual C++ COM compiler support, you can use the _com_raise_error class to report an error, and the _com_error class to retrieve error information. COM will not propagate traditional C++ exceptions as extended IErrorInfo information.

HRESULT Definitions

The return value of COM functions and methods is an HRESULT. The following table lists the standard HRESULT definitions. To use the return values, you must include winerror.h in your project.

HRESULT	Definition
E_NOINTERFACE	The `QueryInterface` function did not recognize the requested interface. The interface is not supported.
E_NOTIMPL	The function contains no implementation.
E_FAIL	An unspecified failure has occurred.
E_OUTOFMEMORY	The function failed to allocate necessary memory.
E_POINTER	Invalid pointer.
E_INVALIDARG	One or more arguments are invalid.
E_UNEXPECTED	A catastrophic failure has occurred.
E_HANDLE	Invalid handle.
E_ABORT	Operation aborted.

Note The information in this section is from the MSDN Library - January 2001, platform SDK:COM (Component Services).

2 Data Types

Minitab Data Types

MtbAppStatusTypes

Defines the different Mtb Application status types.

0 = ASReady (Minitab is ready to accept commands)

1 = ASBusy (Minitab is busy executing a command)

2 = ASError (The last command executed caused an error)

3 = ASQuit (Quit has been called but the application object is not yet destroyed)

MtbDataTypes

Defines the different data types that are currently supported.

0 = Text

1 = Numeric

2 = DateTime

3 = DataUnassigned

MtbFormulaStatusTypes

Defines the state of the Formula for a Column or Constant object.

0 = FSNone

1 = FSUpToDate

2 = FSOutOfDate

3 = FSInvalid

MtbGraphFileTypes

Defines the different graph file types.

0 = GFMinitab

1 = GFJPEG

2 = GFPNGBlackWhite

3 = GFPNGColor

4 = GFPNGHighColor

5 = GFTIFBlackWhite

6 = GFTIFColor

7 = GFBMPBlackWhite

8 = GFBMPColor

9 = GFBMPHighColor

10 = GFGIF

11 = GFEMF

12 = GFSVG

MtbOutputFileTypes

Defines the different Output File types.

0 = OFPlainText

1 = OFHTML

2 = OFRTF

100 = OFDefault

MtbOutputTypes

Defines the different output types allowed in an Output object.

0 = OTGraph

1 = OTTable

2 = OTOutList

3 = OTTitle

4 = OTMessage

5 = OTParagraph

6 = OTFormula

7 = OTTriangle

MtbValueOrderTypes

Defines the display ordering associated with a column.

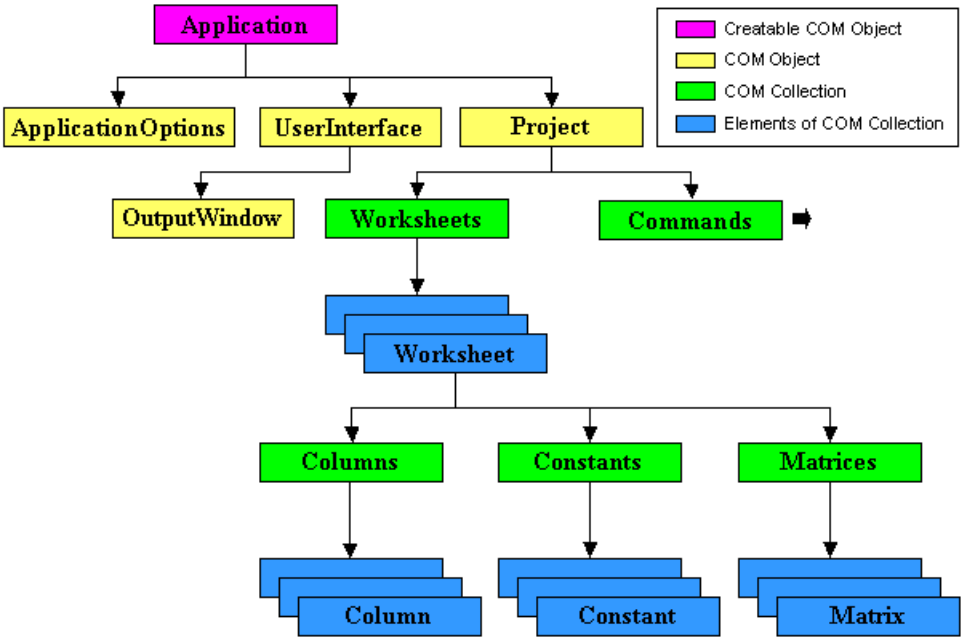
0 = Alphabetical

1 = WorksheetOrder

2 = UserDefined

3 Data Model

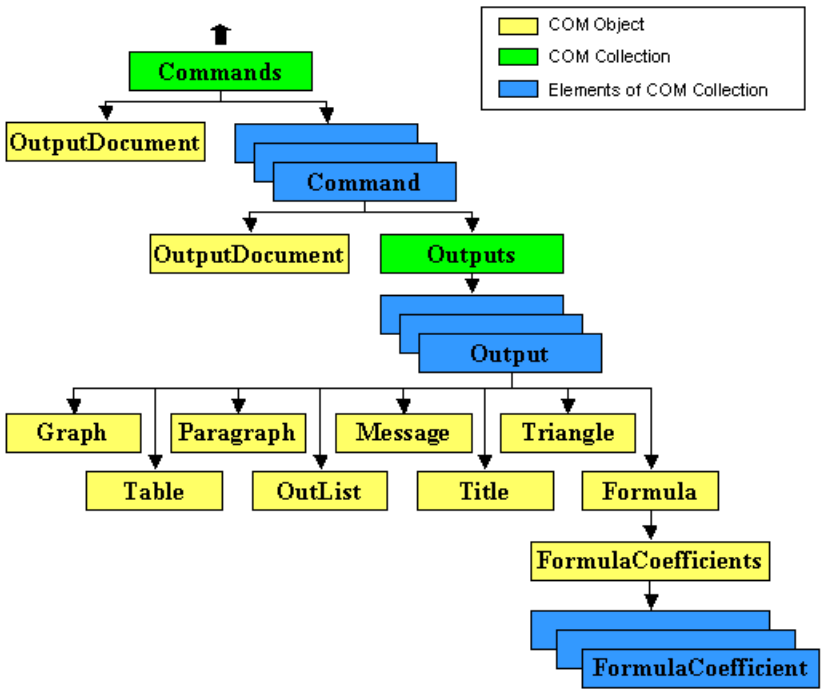
Minitab Automation objects



Object	Description
Application on page 18	The Application object serves as the root node in the Minitab automation server object hierarchy. The Application object is the only object in the hierarchy that can be created by the client. All the lower objects in the hierarchy are accessed through the Application object or through objects contained within the Application object. The Application object provides an interface to allow the client to set and get application-wide global properties.
ApplicationOptions on page 23	Use the `ApplicationOptions` object to read or set options that pertain to the Application object. Use the `Options` property of the `Application` on page 18 object to access `ApplicationOptions`.
UserInterface on page 26	The `UserInterface` object allows control of the Minitab host.
OutputWindow on page 29	The output window in the Minitab application, also called the Session window.
Project on page 31	The `Project` object contains all the information related to an individual project, including the `Worksheets` on page 39 collection and the `Commands` on page 82 collection.
Commands collection on page 82	The `Commands` collection contains the commands that have been issued to Minitab during the session. See Minitab Command Automation Objects on page 10 for the data model.
Worksheets collection on page 39	The Worksheets collection is a set of all the Worksheet objects within a Project object. It supports the standard collection properties and methods.

Object	Description
Worksheet on page 43	The Worksheet object contains all the information related to an individual worksheet, including the `Columns`, `Constants`, and `Matrices` collections, which provide access to all the columns, constants, and matrices in the worksheet.
Columns collection on page 50	The `Columns` collection is a set of all the `Column` objects within a Worksheet object. It supports the standard collection properties and methods.
Column on page 54	The `Column` object contains all the information related to an individual column. The data type on page 7 for each `Column` object can be `Text`, `Numeric`, `DateTime`, or `DataUnassigned`.
Constants collection on page 63	The `Constants` collection is a set of all the `Constant` objects within a `Worksheet` object. It supports the standard collection properties and methods.
Constant on page 67	The `Constant` object contains all the information related to an individual constant. The `Constant` object can contain numeric or text values.
Matrices collection on page 72	The `Matrices` collection is a set of all the `Matrix` objects within a `Worksheet` object. It supports the standard collection properties and methods.
Matrix on page 76	The `Matrix` object contains all the information related to an individual matrix. The `Matrix` object can contain *only* numeric data values.

Minitab Command Automation Objects



Object	Description
Commands collection on page 82	The `Commands` collection contains the commands that have been issued to Minitab during the session.
OutputDocument on page 99	An `OutputDocument` object contains all output generated by a single `Command` on page 85 object or by all commands in the `Commands` on page 82 collection.
Command on page 85	`Command` objects are created when you execute a Minitab command either programmatically or directly in Minitab.
Outputs collection on page 89	The `Outputs` collection for each `Command` on page 85 object contains all the output generated by that command.
Output on page 92	Each `Output` object contains one component of the output from a Minitab Command on page 85 object.
Graph on page 112	Each `Graph` object contains a single graph generated by a Minitab `Command` on page 85 object.
Table on page 105	Each `Table` object contains a single output table generated by a Minitab `Command` on page 85 object.
Paragraph on page 103	Each `Paragraph` object contains a single output paragraph generated by a Minitab `Command` on page 85 object.
OutList on page 114	Each `OutList` object contains a single OutList generated by a Minitab `Command` on page 85 object.
Message on page 117	Each `Message` object contains a single message generated by a Minitab `Command` on page 85 object.
Title on page 102	Each `Title` object contains a single title generated by a Minitab `Command` on page 85 object.
Triangle on page 116	Each `Triangle` object contains a single output triangle generated by a Minitab `Command` on page 85 object.
Formula on page 106	Each `Formula` object contains a single output formula generated by a Minitab `Command` on page 85 object.
FormulaCoefficientsCollection on page 109	The `FormulaCoefficients` collection contains all the coefficients from a formula.
FormulaCoefficient on page 111	Each `FormulaCoefficient` object contains a single coefficient.

4 My Menu

My Menu Overview

By creating specialized dynamic link libraries (DLLs) and placing them in the AddIns folder of your Minitab directory, you can add customized menus to Minitab allowing you to:

- Run a Minitab macro from the menu.
- Display a customized dialog box for running a Minitab macro.
- Launch customized interfaces to corporate databases, or company-created macros.
- Create custom Minitab procedures using Minitab's new COM objects.
- Launch a separate executable from Minitab.

Your custom menus will appear at the right end of the menu bar. Minitab's new customizable menus and toolbars allow you to move any of the items from you custom menus to any menu or toolbar you would like.

My Menu Add-in DLLs

To specify the layout, items, and actions for a custom menu, you need to create a DLL and place it in the AddIns folder of your Minitab directory. An example of such a DLL, along with all supporting files for the Visual Basic project used to create it, are located in the MyMenu folder of your Minitab directory. Portions of the code from the main module of this project are also displayed in My Menu - VB Example on page 13.

Each add-in DLL must include the following 9 procedures:

IMinitabAddin_GetName on page 13() As String
This method returns the friendly name of your add-in.

IMinitabAddin_GetDescription on page 13() As String
This method returns the description of your add-in.

IMinitabAddin_GetMenuItems on page 13(ByRef MainMenu As String, ByRef MenuItems() As String, ByRef Flags As Long)
This method returns the text for the main menu and each menu item. You can return "|" to create a menu separator in your menu items. You must Redim the menu items array to fit your number of items.

IMinitabAddin_OnConnect on page 13(ByVal Hwnd As Long, ByVal App As Object, ByRef Flags As Long)
This method is called as MINITAB is initializing your add-in. The HWND is the handle to the main window. The App Object is a reference to the MINITAB Automation object. You can hold onto either of these for use in your add-in.

IMinitabAddin_OnDisconnect on page 13()
This method is called as Minitab is closing your add-in.

IMinitabAddin_OnDispatchCommand on page 13(ByVal Menu As Long) As String
This method is called whenever a user selects one of your menu items. The Menu should be equivalent to the menu item index set in GetMenuItems.

IMinitabAddin_OnNotify on page 13(ByVal NotifyType As AddinNotifyType)
This method is called when Minitab notifies your add-in that something has changed. Use the NotifyType to figure out what changed.

IMinitabAddin_QueryCustomCommand on page 13(ByVal Cmnd As String) As Boolean
This method is called when Minitab asks your Addin if it supports a custom command. Return True if you support the command.

IMinitabAddin_ExecuteCustomCommand on page 13(ByVal Cmnd As String, ByRef Args() As String)
This method is called when Minitab asks your Addin to execute a custom command. The Cmnd is the name of the command and Args is an array of arguments.

My Menu - VB Example

```
' Your add-in must implement the add-in interface
' Did you remember to add the Minitab 15.0 Addin Interface to the project references?
' Also, if you want to use the Minitab objects, remember to add a reference to the
'   Mtb 15.0 Type Library

Implements IMinitabAddin

Private Function IMinitabAddin_GetName() As String
  ' This method returns the friendly name of your add-in
  ' Both the name and the description of the add-in are stored in the registry.
  IMinitabAddin_GetName = "Minitab My Menu Add-In"
End Function

Private Function IMinitabAddin_GetDescription() As String
  ' This method returns the description of your add-in
  IMinitabAddin_GetDescription = "Example add-in using the My Menu functionality"
End Function

Private Sub IMinitabAddin_GetMenuItems(ByRef MainMenu As String,
                                       ByRef MenuItems() As String, ByRef Flags As Long)
  ' This method returns the text for the main menu and each menu item
  ' You can return "|" to create a menu separator in your menu items
  ' You must Redim the menu items array to fit your number of items
  MainMenu = "&My Menu"
  ReDim MenuItems(0 To 4)
  MenuItems(0) = "Describe &column(s)..."
  MenuItems(1) = "Rename active &worksheet..."
  MenuItems(2) = "|"
  MenuItems(3) = "&DOS window"
  MenuItems(4) = "&Geometric Mean and Mean Absolute Difference..."
  ' Flags is not currently used
  Flags = 0
End Sub

Private Sub IMinitabAddin_OnConnect(ByVal Hwnd As Long, ByVal App As Object,
                                    ByRef Flags As Long)
  ' This method is called as Minitab is initializing your add-in
  ' The HWND is the handle to the main Minitab window
  ' The App object is a reference to the Minitab Automation object
  ' You can hold onto either of these for use in your add-in
  ' Flags is used to tell Minitab if your add-in has dynamic menus (i.e. should be
  ' reloaded each time Minitab starts up). Set Flags to 1 for dynamic menus and 0
  ' for static.
  Set gMTBApp = App
  ' This forces Minitab to retain all commands (even those run by the interactive user).
  gMTBApp.Options.SaveCommands = True
  ' Static menus
  Flags = 0
```

```
End Sub

Private Sub IMinitabAddin_OnDisconnect()
  ' This method is called as Minitab is closing your add-in
  Set gMTBApp = Nothing
End Sub

Private Function IMinitabAddin_OnDispatchCommand(ByVal Menu As Long) As String
  ' This method is called whenever a user selects one of your menu items
  ' The Menu should be equivalent to the menu item index set in GetMenuItems
  Dim Cmnd As String
  Select Case Menu
    Case 0
      ' Describe column(s)
      Dim lColCt As Long
      ' Fill up list box in dialog with numeric columns in worksheet
      FormDescribe.lstColumns.Clear
      lColCt = gMTBApp.ActiveProject.ActiveWorksheet.Columns.Count
      For i = 1 To lColCt
        ' Only select the numeric columns.
        If gMTBApp.ActiveProject.ActiveWorksheet.Columns.Item(i).DataType = Numeric Then
          FormDescribe.lstColumns.AddItem
            gMTBApp.ActiveProject.ActiveWorksheet.Columns(i).SynthesizedName
        End If
      Next
      ' Show the dialog
      FormDescribe.Show 1
      ' The Minitab command to issue
      If FormDescribe.Tag = "OK" Then
        Dim sTemp As String
        Dim bPrev As Boolean
        bPrev = False
        sTemp = "Describe "
        For i = 1 To FormDescribe.lstColumns.ListCount
          FormDescribe.lstColumns.ListIndex = i - 1
          If FormDescribe.lstColumns.Selected(i - 1) Then
            If bPrev Then
              sTemp = sTemp & " "
            End If
            sTemp = sTemp & FormDescribe.lstColumns.Text
            bPrev = True
          End If
        Next
        If FormDescribe.chkMean.Value = 1 Then
          sTemp = sTemp & ";  Mean"
        End If
        If FormDescribe.chkVariance.Value = 1 Then
          sTemp = sTemp & ";  Variance"
        End If
        If FormDescribe.chkSum.Value = 1 Then
          sTemp = sTemp & ";  Sums"
        End If
        If FormDescribe.chkNnonmissing.Value = 1 Then
          sTemp = sTemp & ";  N"
        End If
        If FormDescribe.chkHistogram.Value = 1 Then
          sTemp = sTemp & ";  GHist"
        End If
        If FormDescribe.chkBoxplot.Value = 1 Then
          sTemp = sTemp & ";  GBoxplot"
        End If
```

```
      sTemp = sTemp & "."
      ' If you simply want to run a Minitab command you can either set that string
      ' as the return value for this function, or you can call ExecuteCommand
      ' gMTBApp.ActiveProject.ExecuteCommand sTemp
      Cmnd = sTemp
    End If
    Unload FormDescribe
  Case 1
    ' Rename active worksheet
    Dim sCurrent As String
    sCurrent = gMTBApp.ActiveProject.ActiveWorksheet.Name
    FormRename.txtCurrent.Enabled = True
    FormRename.txtCurrent.Text = sCurrent
    FormRename.txtCurrent.Enabled = False
    ' Show the dialog
    FormRename.Show 1
    If FormRename.Tag = "OK" Then
      gMTBApp.ActiveProject.ActiveWorksheet.Name = FormRename.txtNew.Text
    End If
    Unload FormRename
  Case 3
    ' DOS Window
    Dim sSysDir As String
    Dim sWinDir As String
    Dim lLength As Long
    Dim sDOSPath As String
    Dim DOSID
    Dim hsearch As Long
    Dim findinfo As WIN32_FIND_DATA
    ' Allocate string
    sSysDir = Space(255)
    sWinDir = Space(255)
    ' Get path of system directory
    lLength = GetSystemDirectory(sSysDir, 255)
    ' Trim blank space from string
    sSysDir = Left(sSysDir, lLength)
    lLength = GetWindowsDirectory(sWinDir, 255)
    sWinDir = Left(sWinDir, lLength)
    sDOSPath = sSysDir & "\cmd.exe"
    hsearch = FindFirstFile(sDOSPath, findinfo)
    ' WinME uses command.com
    ' To be on the safe side, look around for DOS before
    ' giving up.
    If hsearch = -1 Then
      sDOSPath = sSysDir & "\command.com"
      hsearch = FindFirstFile(sDOSPath, findinfo)
      If hsearch = -1 Then
        sDOSPath = sWinDir & "\cmd.exe"
        hsearch = FindFirstFile(sDOSPath, findinfo)
        If hsearch = -1 Then
          sDOSPath = sWinDir & "\command.com"
          hsearch = FindFirstFile(sDOSPath, findinfo)
        End If
      End If
    End If
    ' Open DOS window
    If hsearch = -1 Then
      MsgBox "Cannot locate DOS executable", vbOKOnly, "MyMenu"
    Else
      FindClose (hsearch)
      DOSID = Shell(sDOSPath, 1)
```

```
      End If
    Case 4
      ' Geometric Mean and Mean Absolute Difference (stored in the worksheet)
      ' Fill up list box in dialog with numeric columns in worksheet
      lColCt = gMTBApp.ActiveProject.ActiveWorksheet.Columns.Count
      For i = 1 To lColCt
        If gMTBApp.ActiveProject.ActiveWorksheet.Columns(i).DataType = Numeric Then
          Dim sColName As String
          lNumber = gMTBApp.ActiveProject.ActiveWorksheet.Columns(i).Number
          sName = gMTBApp.ActiveProject.ActiveWorksheet.Columns(i).SynthesizedName
          ' Add column name (if it exists)
          sColName = gMTBApp.ActiveProject.ActiveWorksheet.Columns(i).Name
          If sColName <> sName Then
            sName = sName & "  "
            sName = sName & sColName
          End If
          FormGeoMean.Combo1.AddItem sName
          FormGeoMean.Combo2.AddItem Str(i)
        End If
      Next
      ' Show the dialog
      FormGeoMean.Show 1
      If FormGeoMean.Tag = "OK" Then
        ' Get data from column and pass it to function to do calculations
        Dim MTBColumn As Mtb.Column
        Set MTBColumn =
          gMTBApp.ActiveProject.ActiveWorksheet.Columns(CLng(FormGeoMean.Combo2.Text))
        Dim dData() As Double
        ReDim dData(MTBColumn.RowCount)
        dData = MTBColumn.GetData()
        Dim dGeoMean As Double
        Dim bSuccess As Boolean
        ' FindGeoMean takes an array of doubles and returns the geometric mean.
        ' bSuccess indicates if the calculations were completed.
        dGeoMean = FindGeoMean(dData, bSuccess)
        If bSuccess = True Then
          ' Mean Absolute Difference
          Dim dMAD As Double
          dMAD = FindMAD(dData)
          ' Store both values in the first available column.
          Dim StorageCol As Mtb.Column
          Set StorageCol = gMTBApp.ActiveProject.ActiveWorksheet.Columns.Add
          StorageCol.SetData dGeoMean, 1, 1
          StorageCol.SetData dMAD, 2, 1
          StorageCol.Name = "MyResults"
        Else
          ' An error occurred.
          gMTBApp.ActiveProject.ExecuteCommand "NOTE ** Error ** Cannot compute statistics"

        End If
      End If
      Unload FormGeoMean
  End Select
  IMinitabAddin_OnDispatchCommand = Cmnd
End Function

Private Sub IMinitabAddin_OnNotify(ByVal NotifyType As AddinNotifyType)
  ' This method is called when Minitab notifies your add-in that something has changed.
  ' Use the NotifyType to figure out what changed
  ' Minitab currently fires no events, so this method is not called.
End Sub
```

```
Private Function IMinitabAddin_QueryCustomCommand(ByVal Cmnd As String) As Boolean
  ' This method is called when Minitab asks your Addin if it supports a custom command
  ' The Cmnd is the name of the custom command.  Return True if you support the command
  If UCase(Cmnd) = "EXPLORER" Or UCase(Cmnd) = "CLEAR" Then
    IMinitabAddin_QueryCustomCommand = True
  Else
    IMinitabAddin_QueryCustomCommand = False
  End If
End Function

Private Sub IMinitabAddin_ExecuteCustomCommand(ByVal Cmnd As String,
                                               ByRef Args() As String)
  ' This method is called when Minitab asks your Addin to execute a custom command
  ' The Cmnd is the name of the command and Args is an array of arguments
  If UCase(Cmnd) = "EXPLORER" Then
    ' Open Windows Explorer
    Dim sWinDir As String
    Dim lLength As Long
    Dim sExplorerPath As String
    Dim ExplorerID
    ' Allocate string
    sWinDir = Space(255)
    ' Get path of windows directory
    lLength = GetWindowsDirectory(sWinDir, 255)
    ' Trim blank space from string
    sWinDir = Left(sWinDir, lLength)
    sExplorerPath = sWinDir & "\explorer.exe"
    ' Open explorer window
    ExplorerID = Shell(sExplorerPath, 1)
    AppActivate ExplorerID
  ElseIf UCase(Cmnd) = "CLEAR" Then
    ' Clear indicated columns
    Dim lColCt As Long
    lColCt = gMTBApp.ActiveProject.ActiveWorksheet.Columns.Count()
    For Each Arg In Args
      For i = 1 To lColCt
        If gMTBApp.ActiveProject.ActiveWorksheet.Columns.Item(i).Number = Arg Then
          gMTBApp.ActiveProject.ActiveWorksheet.Columns.Item(i).Clear
        End If
      Next
    Next
  End If
End Sub
```

A Minitab Automation Object Reference

Application object

The Application object serves as the root node in the Minitab automation server object hierarchy. The Application object is the only object in the hierarchy that can be created by the client. All the lower objects in the hierarchy are accessed through the Application object or through objects contained within the Application object. The Application object provides an interface to allow the client to set and get application-wide global properties.

Properties

Property	Description
ActiveProject on page 19	Returns the currently active project.
AppPath on page 19	The path to the currently running Minitab executable.
Handle on page 19	The handle to the main Minitab window.
LastError on page 20	Holds the last error that occurred during execution of a command.
Options on page 20	Returns the `ApplicationOptions` object.
Status on page 20	Indicates the current status of the Minitab application.
UserInterface on page 21	Returns the `UserInterface` object of the application.

Methods

Method	Description
Help on page 21	Use to launch this Help file.
New on page 21	Use to create a new project and make it the active project.
Open on page 22	Use to open an existing project file and make it the active project.
Quit on page 22	Use to close and delete all objects in the `Application` object hierarchy, including the `Application` object.

Example

Create a Minitab Application object (`MtbApp`) and make it visible to the user. Then display a message box with the values of the `Status`, `LastError`, `AppPath`, and `Handle` properties, as well as the `DefaultFilePath` property of the `ApplicationOptions` object. Finally, change the comment for the active project via the `ActiveProject` property.

```
Dim MtbApp As New Mtb.Application

With MtbApp
    .UserInterface.Visible = True

    MsgBox ( _
        "Status = " & .Status & vbCrLf & _
        "LastError = " & .LastError & vbCrLf & _
        "Default File Path = " & .Options.DefaultFilePath & vbCrLf & _
        "Application Path = " & .AppPath & vbCrLf & _
        "Window Handle = " & .Handle)
```

```
    .ActiveProject.Comment = "New Minitab Project."
End With
```

Application property - ActiveProject

Description
Returns the currently active project.

Type
Project

Range
N/A

Access
Read-only

Application property - AppPath

Description
The path to the currently running Minitab executable.

Type
String

Range
Valid string

Access
Read-only

Application property - Handle

Description
The handle to the main Minitab window.

Type
Long

Range
Any valid long integer

Access
Read-only

Application property - LastError

Description
Holds the last error that occurred during execution of a command.

Type
String

Range
Valid string

Access
Read-only

After executing an asynchronous command, the `Status` on page 20 property should be checked to see when the command completes. If the `Status` property indicates an error occurred then use the `LastError` property to retrieve the error message.

Application property - Options

Description
Returns the `ApplicationOptions` object.

Type
ApplicationOptions on page 23

Range
N/A

Access
Read-only

Application property - Status

Description
Indicates the current status of the Minitab application.

Type
MtbAppStatusTypes on page 7

Range
Any `MtbAppStatusTypes` constant

Access
Read-only

After executing an asynchronous command, the `Status` property should be checked to see when the command completes. If the `Status` property indicates an error occurred then use the `LastError` on page 20 property to retrieve the error message.

Application property - UserInterface

Description
Returns the `UserInterface` object of the application.

Type
UserInterface on page 26

Range
N/A

Access
Read-only

Application method - Help

Use to launch this Help file.

Syntax

```
Help()
```

Returns

```
HRESULT
```

Example

Call the online Help file.

```
Dim MtbApp As Mtb.Application
Set MtbApp = New Mtb.Application

MtbApp.Help
```

Application method - New

Use to create a new project and make it the active project.

Syntax

```
New()
```

Returns

```
HRESULT
```

Example

Create a new project and make it the active project.

```
Dim MtbApp As Mtb.Application
Set MtbApp = New Mtb.Application
```

```
MtbApp.New
```

Application method - Open

Use to open an existing project file and make it the active project.

Syntax

```
Open(Filename as String)
```

Arguments

Filename

Required. The path and name of the project file to be opened. If a path is not specified, the DefaultFilePath on page 24 is used.

Returns

```
HRESULT
```

Example

Open an existing project file and make it the active project.

```
Dim MtbApp As Mtb.Application
Set MtbApp = New Mtb.Application

MtbApp.Open "C:\MyProject.mpj"
```

Application method - Quit

Use to close and delete all objects in the `Application` object hierarchy, including the `Application` object.

Syntax

```
Quit()
```

Returns

```
HRESULT
```

Example

Delete all the objects in the `Application` Object hierarchy, including `MtbApp`.

```
Dim MtbApp As Mtb.Application
Set MtbApp = New Mtb.Application

MtbApp.Quit
```

ApplicationOptions object

Use the `ApplicationOptions` object to read or set options that pertain to the Application object. Use the `Options` property of the `Application` on page 18 object to access `ApplicationOptions`.

Properties

Property	Description
ClientMissingValueDateTime on page 23	Date/time to use to represent missing date/time values when receiving date/time values from a client or giving date/time values to a client.
ClientMissingValueNumeric on page 24	Number to use to represent missing numeric values when receiving numeric data from a client or giving numeric data to a client.
DefaultFilePath on page 24	Default file path used by the application for opening/saving files.
DefaultOutputFileType on page 25	Returns or sets the default type of output document file that will be produced.
SaveCommands on page 26	If False, commands created directly in Minitab via Session window or menu commands are automatically deleted from the Commands collection, and only commands created with ExecuteCommand or ExecuteCommandAsync are saved.

Example

Create a Minitab Application object (`MtbApp`), then use the `ApplicationOptions` on page 23 object to access and display the `DefaultFilePath` in a message box.

```
Dim MtbApp As Mtb.Application
Dim MtbAppOpt As Mtb.ApplicationOptions

Set MtbApp = New Mtb.Application
Set MtbAppOpt = MtbApp.Options

MsgBox "The current default file path is " & _
    MtbAppOpt.DefaultFilePath
```

ApplicationOptions property - ClientMissingValueDateTime

Description

Date/time to use to represent missing date/time values when receiving date/time values from a client or giving date/time values to a client.

Type

Date

Range

Valid COM DATE value

Default

12/31/9999

Access

Read/Write

This property does not affect Minitab's convention that uses "*" to represent missing values; therefore, in the Minitab worksheet missing values will always appear as "*".

Example

Set the `ClientMissingValueDateTime` to June1, 1990 for a Minitab Application object, then display it in a message box.

```
Dim MtbApp As Mtb.Application
Dim MtbAppOpt As Mtb.ApplicationOptions

Set MtbApp = New Mtb.Application
Set MtbAppOpt = MtbApp.Options

MtbAppOpt.ClientMissingValueDateTime = #6/1/1990#
MsgBox "The ClientMissingValueDateTime is " _
    & MtbAppOpt.ClientMissingValueDateTime
```

ApplicationOptions property - ClientMissingValueNumeric

Description

Number to use to represent missing numeric values when receiving numeric data from a client or giving numeric data to a client.

Type

Double

Range

Valid double precision value

Default

1.23456E30

Access

Read/Write

This property does not affect Minitab's convention that uses "*" to represent missing values; therefore, in the Minitab worksheet missing numeric values will always appear as "*".

Example

Set the `ClientMissingValueNumeric` to 0.001 for a Minitab Application object, then display it in a message box.

```
Dim MtbApp As Mtb.Application
Dim MtbAppOpt As Mtb.ApplicationOptions

Set MtbApp = New Mtb.Application
Set MtbAppOpt = MtbApp.Options

MtbAppOpt.ClientMissingValueNumeric = 0.001
MsgBox "The ClientMissingValueNumeric is " & _
    MtbAppOpt.ClientMissingValueNumeric
```

ApplicationOptions property - DefaultFilePath

Description

Default file path used by the application for opening/saving files.

Type

String

Range
Any valid path

Access
Read/Write

The default `DefaultFilePath` is the directory where the task scheduler will schedule the task to execute.

Example

Create a Minitab Application object (`MtbApp`), then use the `ApplicationOptions` on page 23 object to access and display the `DefaultFilePath` in a message box.

```
Dim MtbApp As Mtb.Application
Dim MtbAppOpt As Mtb.ApplicationOptions

Set MtbApp = New Mtb.Application
Set MtbAppOpt = MtbApp.Options

MsgBox "The current default file path is " & _
    MtbAppOpt.DefaultFilePath
```

ApplicationOptions property - DefaultOutputFileType

Description
Returns or sets the default type of output document file that will be produced.

Type
MtbOutputFileTypes on page 8

Range
Any `MtbOutputFileTypes` constant

Access
Read/Write

The default is HTML.

Example

Create a Minitab Application object (`MtbApp`), set the `DefaultOutputFileType` to RTF, then display the `DefaultOutputFileType` in a message box.

```
Dim MtbApp As Mtb.Application
Dim MtbAppOpt As Mtb.ApplicationOptions

Set MtbApp = New Mtb.Application
Set MtbAppOpt = MtbApp.Options

MtbAppOpt.DefaultOutputFileType = OFRTF

MsgBox "The current default output file type is " & _
    MtbAppOpt.DefaultOutputFileType
```

ApplicationOptions property - SaveCommands

Description

If False, commands created directly in Minitab via Session window or menu commands are automatically deleted from the Commands collection, and only commands created with ExecuteCommand or ExecuteCommandAsync are saved. The exception is that most commands that produce graphs are not deleted until the graphs are closed.

Type

Boolean

Range

True/False

Access

Read/Write

The default is True.

Example

Create a Minitab Application object (MtbApp), then set `SaveCommands` to `False`.

```
Dim MtbApp As Mtb.Application
Dim MtbAppOpt As Mtb.ApplicationOptions

Set MtbApp = New Mtb.Application
Set MtbAppOpt = MtbApp.Options

MsgBox ("SaveCommands = " & MtbAppOpt.SaveCommands)
MtbAppOpt.SaveCommands = False
```

UserInterface object

The `UserInterface` object allows control of the Minitab host.

Properties

Property	Description
ActivePrinter on page 27	The name of the active printer.
DisplayAlerts on page 27	Controls whether Minitab displays alerts and messages while a client script is running.
Interactive on page 28	Controls whether the user is able to issue commands or alter data directly in Minitab if it is visible.
OutputWindow on page 28	Returns the `OutputWindow` object of the `UserInterface` object.
UserControl on page 28	Controls whether Minitab quits when the last client object is released.
Visible on page 29	Controls whether Minitab server is visible.

Example

Create a Minitab Application object (`MtbApp`), make it visible to the user, and set the `Interactive`, `UserControl`, and `DisplayAlerts` properties of the `UserInterface` object to `True`. Then display a message box with the value of the `ActivePrinter` property.

```
Dim MtbApp As New Mtb.Application

With MtbApp.UserInterface
    .Visible = True
    .Interactive = True
    .UserControl = True
    .DisplayAlerts = True
    MsgBox ("ActivePrinter = " & .ActivePrinter)
End With
```

UserInterface property - ActivePrinter

Description

The name of the active printer.

Type

String

Range

Valid string

Access

Read/Write

By default, the default system printer is active. Typically, you can specify a different printer by doing one of the following:

- For a For a Windows network printer, set the `ActivePrinter` property equal to the path of the network printer. For example, `App.UserInterface.ActivePrinter = "\\Printserver1\PD 4000 PS"`.
- For a Windows network printer, set the `ActivePrinter` property equal to the name of the printer as it appears in the **Control Panel** > **Printers** folder. For example, `App.UserInterface.ActivePrinter = "Epson Stylus COLOR 500 ESC/P 2"`.

UserInterface property - DisplayAlerts

Description

Controls whether Minitab displays alerts and messages while a client script is running.

Type

Boolean

Range

True/False

Access

Read/Write

It is good practice to set `DisplayAlerts` back to `True` when a script finishes. Minitab does not do this automatically.

UserInterface property - Interactive

Description
Controls whether the user is able to issue commands or alter data directly in Minitab if it is visible.

Type
Boolean

Range
True/False

Access
Read/Write

The default is True.

UserInterface property - OutputWindow

Description
Returns the `OutputWindow` object of the `UserInterface` object.

Type
OutputWindow on page 29

Range
N/A

Access
Read-only

Example

Create a new instance of Minitab, set the `CommandsEnabled` property of the `OutputWindow` object to `False`, then write a title and some text to the Session window using the `WriteTitle` and `WriteText` methods.

```
Dim MtbApp As New Mtb.Application

With MtbApp.UserInterface
    .Visible = True
    With .OutputWindow
        .CommandsEnabled = False
        .WriteTitle ("This is my title for the Session window")
        .WriteText ("This is my text for the Session window.")
    End With
End With
```

UserInterface property - UserControl

Description
Controls whether Minitab quits when the last client object is released.

Type
Boolean

Range
True/False

Access
Read/Write

The default is `False` if the application was started programmatically. Minitab must be visible for `UserControl` to be `True`.

UserInterface property - Visible

Description
Controls whether Minitab server is visible.

Type
Boolean

Range
True/False

Access
Read/Write

The default is `True` when Minitab is started by the user, `False` when started programmatically.

OutputWindow object

The output window in the Minitab application, also called the Session window.

Properties

Property	Description
CommandsEnabled on page 30	If `True`, the command prompt is enabled in the application's output (Session) window.

Methods

Method	Description
WriteText on page 30	Use to display text in Minitab's Session window.
WriteTitle on page 30	Use to display text in Minitab's Session window.

Example

Create a new instance of Minitab, set the `CommandsEnabled` property of the `OutputWindow` object to `False`, then write a title and some text to the Session window using the `WriteTitle` and `WriteText` methods.

```
Dim MtbApp As New Mtb.Application

With MtbApp.UserInterface
    .Visible = True
    With .OutputWindow
```

```
        .CommandsEnabled = False
        .WriteTitle ("This is my title for the Session window")
        .WriteText ("This is my text for the Session window.")
    End With
End With
```

OutputWindow property - CommandsEnabled

Description

If `True`, the command prompt is enabled in the application's output (Session) window.

Type

Boolean

Range

True/False

Access

Read/Write

OutputWindow method - WriteText

Use to display text in Minitab's Session window.

Syntax

```
WriteText(Text as String)
```

Arguments

Text

Required. Text to write to the Session window.

Returns

```
HRESULT
```

Remarks

Text is displayed in the current I/O font specified in the Minitab application.

OutputWindow method - WriteTitle

Use to display text in Minitab's Session window.

Syntax

```
WriteTitle(Title as String)
```

Arguments

Title
Required. Text to write to the Session window.

Returns

HRESULT

Remarks

The title is displayed in the current title font specified in the Minitab application.

Project object

The `Project` object contains all the information related to an individual project, including the `Worksheets` on page 39 collection and the `Commands` on page 82 collection.

Properties

Property	Description
ActiveWorksheet on page 32	Returns or sets the active worksheet for the project.
Commands on page 33	Returns the `Commands` on page 82 collection for the project.
Comment on page 33	Comment for the project.
Creator on page 33	Creator of the project.
Date on page 34	Date of the project.
FullName on page 34	Full name of the `Project` object disk file, including the path name and/or drive name, set when a project is opened or saved.
Name on page 34	Name of the project and its disk file.
Path on page 35	Path of the `Project` object disk file, set when a project is opened or saved
Worksheets on page 35	Returns the `Worksheets` on page 39 collection of the project.

Methods

Method	Description
CancelCommand on page 36	Use to cancel the execution of a user-issued or COM-issued command.
Delete on page 36	Use to delete the `Project` object and the underlying `Worksheets` on page 39 and `Commands` on page 82 collections.
ExecuteCommand on page 37	Use to run a Minitab session command and create a command object.
ExecuteCommandAsync on page 37	Use to run a Minitab session command asynchronously and create a command object.
Save on page 38	Use to save the project to `FullName` on page 34.
SaveAs on page 38	Use to save a copy of the project.

Example

Create a Minitab Application object, execute Minitab commands both synchronously and asynchronously, and then attempt to cancel the asynchronous command. Also use the `Commands`, `Comment`, `Creator`, and `Date` properties as well as the `Save` and `SaveAs` methods.

```
Dim MtbApp As New mtb.Application
Dim MtbProj As mtb.Project
Dim Status As MtbAppStatusTypes

Set MtbProj = MtbApp.ActiveProject

MtbProj.ExecuteCommand ("RAND 30 C1")
MtbProj.ExecuteCommandAsync ("RAND 100000 c2-c100")
    Status = MtbApp.Status
    If (Status = ASBusy) Then
        MtbProj.CancelCommand
    End If

MtbApp.UserInterface.Visible = True

'Save the project as a Release 14 project called MyProject
MtbProj.SaveAs "C:\MyProject", True, 14

'Add creator, date, and comment information
MtbProj.Creator = "Me"
MtbProj.Date = Date
MtbProj.Comment = "This is my project."

'Dislplay creator, date, and comment information
MsgBox "This project created by " & MtbProj.Creator _
    & " on " & MtbProj.Date _
    & vbCrLf & "Comment: " & MtbProj.Comment

'Save the project again
MtbProj.Save
```

Project property - ActiveWorksheet

Description

Returns or sets the active worksheet for the project.

Type

Worksheet on page 43

Range

N/A

Access

Read/Write

Example

Create a Minitab `Application` on page 18 object, then rename the project's active worksheet and display a message with the new name.

```
Dim MtbApp As Mtb.Application
Dim MtbSheet As Mtb.Worksheet

Set MtbApp = New Mtb.Application
Set MtbSheet = MtbApp.ActiveProject.ActiveWorksheet

MtbSheet.Name = "My Worksheet"
```

```
MsgBox "The active worksheet is called " & _
    MtbApp.ActiveProject.ActiveWorksheet.Name
```

Project property - Commands

Description
Returns the Commands on page 82 collection for the project.

Type
Commands on page 82 collection

Range
N/A

Access
Read-only

Project property - Comment

Description
Comment for the project.

Type
String

Range
Valid string

Access
Read/Write

Default is blank.

Project property - Creator

Description
Creator of the project.

Type
String

Range
Valid string

Access
Read/Write

Default is blank.

Project property - Date

Description
Date of the project.

Type
String

Range
Valid string

Access
Read/Write

Default is blank

Project property - FullName

Description
Full name of the `Project` object disk file, including the path name and/or drive name, set when a project is opened or saved.

Type
String

Range
Valid file name, including the path name and/or drive name

Access
Read-only

Example

Display the `FullName` property (path and name) of the project in a message box.

```
Dim MtbApp As Mtb.Application
Dim MtbProj As Mtb.Project

Set MtbApp = New Mtb.Application
Set MtbProj = MtbApp.ActiveProject

MtbProj.Name = "My Project"
MsgBox "The FullName is " & _
    MtbProj.FullName
```

Project property - Name

Description
Name of the project and its disk file. It also is the name of the file if the project is saved to disk. Setting the `Name` property automatically updates the file name portion of the `FullName` property.

Type
String

Range
Any valid file name

Access
Read/Write

Do not include the path when setting the `Name` property.

Example

Retrieve the active project, name it "My Project," and print the name in a message box.

```
Dim MtbApp As Mtb.Application
Dim MtbProj As Mtb.Project

Set MtbApp = New Mtb.Application
Set MtbProj = MtbApp.ActiveProject

MtbProj.Name = "My Project"
MsgBox "The project is " & MtbProj.Name
```

Project property - Path

Description
Path of the `Project` object disk file, set when a project is opened or saved

Type
String

Range
Valid path name. It may include the drive name.

Access
Read-only

Example

Display the `Path` property of the project in a message box.

```
Dim MtbApp As Mtb.Application
Dim MtbProj As Mtb.Project

Set MtbApp = New Mtb.Application
Set MtbProj = MtbApp.ActiveProject

MsgBox "The Path is " & _
    MtbProj.Path
```

Project property - Worksheets

Description
Returns the `Worksheets` on page 39 collection of the project.

Type
Worksheets on page 39 collection

Range
N/A

Access
Read-only

Example

Retrieve the `Worksheets` on page 39 collection from the `Project` object and display a message with the number of worksheets in the collection.

```
Dim MtbApp As Mtb.Application
Dim MtbProj As Mtb.Project
Dim MtbSheets As Mtb.Worksheets

Set MtbApp = New Mtb.Application
Set MtbProj = MtbApp.ActiveProject
Set MtbSheets = MtbProj.Worksheets

MsgBox "There are this many worksheets in the project: " & _
    MtbSheets.Count
```

Project method - CancelCommand

Use to cancel the execution of a user-issued or COM-issued command.

Syntax

```
CancelCommand()
```

Returns

```
Boolean
```

Remarks

Returns `True` if the command was cancelled, `False` if no command was executing.

Project method - Delete

Use to delete the `Project` object and the underlying `Worksheets` on page 39 and `Commands` on page 82 collections. `Delete` also sets the active project for the application to NULL.

Syntax

```
Delete()
```

Returns

```
HRESULT
```

Example

Delete the `Project` object and the underlying `Worksheets` on page 39 and `Commands` on page 82 collections.

```
Dim MtbApp As Mtb.Application
Dim MtbProj As Mtb.Project

Set MtbApp = New Mtb.Application
Set MtbProj = MtbApp.ActiveProject

MtbProj.Delete
```

Project method - ExecuteCommand

Use to run a Minitab session command and create a command object.

Syntax

```
ExecuteCommand(Command as String, WorksheetObj as Worksheet)
```

Arguments

Command

Required. One or more session commands to execute in Minitab. Multiple commands and subcommands may be included in the same command. Subcommands must be separated from commands and from each other by semicolons. Each command must end with a period, with the exception of LET. LET commands must be separated from other commands by a new line rather than a period.

WorksheetObj

Optional. The worksheet to use when executing the command. The specified worksheet becomes the `ActiveWorksheet`. If none specified, then the current `ActiveWorksheet` is used.

Remarks

The command is executed synchronously, meaning this interface will not return until the command has completed executing, giving direct feedback as to the completion status of the command, success or failure.

Project method - ExecuteCommandAsync

Use to run a Minitab session command asynchronously and create a command object.

Syntax

```
ExecuteCommandAsync(Command as String, WorksheetObj as Worksheet)
```

Arguments

Command

Required. One or more session commands to execute in Minitab. Multiple commands and subcommands may be included in the same command. Subcommands must be separated from commands and from each other by semicolons. Each command must end with a period, with the exception of LET. LET commands must be separated from other commands by a new line rather than a period.

WorksheetObj
Optional. The worksheet to use when executing the command. The specified worksheet becomes the `ActiveWorksheet`. If none specified, then the current `ActiveWorksheet` is used.

Returns

`HRESULT`

Remarks

The command is submitted for execution asynchronously, meaning this interface will return before the command is executed. Use the `Application` on page 18 object's `Status` on page 20 property to see if the command completed successfully. If an error occurred, use the application's `LastError` on page 20 property to retrieve the error message.

Project method - Save

Use to save the project to `FullName` on page 34.

Syntax

```
Save()
```

Returns

`HRESULT`

Project method - SaveAs

Use to save a copy of the project.

Syntax

```
SaveAs(Filename as String, Replace as Boolean, Version as Long)
```

Arguments

Filename
Optional. Path and file name to use when saving the project. If a path is not specified, then the `DefaultFilePath` on page 24 is used. If a file name is not specified then the `Name` on page 34 property is used.

Replace
Optional. If `True`, an existing file with the same name will be overwritten. The default is `True`.

Version
Optional. The Minitab version number to save the project as. If not specified, the current version number is used.

Returns

`HRESULT`

B Worksheet Object Reference

Worksheets Collection object

The Worksheets collection is a set of all the Worksheet objects within a Project object. It supports the standard collection properties and methods.

Properties

Property	Description
Count on page 39	Number of `Worksheet` objects within the `Worksheets` collection.

Methods

Method	Description
Add on page 40	Use to add *`Count`* `Worksheet` objects to the `Worksheets` collection.
Delete on page 40	Use to remove all `Worksheet` objects from the `Worksheets` collection.
Item on page 41	Use to return a `Worksheet` object within the `Worksheets` collection.
Open on page 42	Use to open an existing Minitab worksheet disk file, read it into a `Worksheet` object, and add the `Worksheet` object to the end of the `Worksheets` collection.
Remove on page 42	Use to delete a `Worksheet` object and remove it from the `Worksheets` collection.

Example

Retrieve the Worksheets collection from the Project , add two worksheets to it and name the first one "First Year," open an existing Minitab worksheet ("Market"), then remove the second worksheet from the Worksheets collection:

```
Dim MtbSheets As Mtb.Worksheets
Set MtbSheets = MtbProject.Worksheets
MtbSheets.Add(2).Name = "First Year"
MtbSheets.Open "Market"
MtbSheets.Remove (2)
```

Note This example assumes that the `MtbProject` object was previously initialized to a valid `Project` object as demonstrated in the Project object example.

Worksheets Collection property - Count

Description

Number of `Worksheet` objects within the `Worksheets` collection.

Type

Long

Range

0 - number of `Worksheet` objects within the `Worksheets` collection

Access
Read-only

Example

Retrieve the Worksheets collection from a previously initialized Project object, then display the number of Worksheet objects in the Worksheets collection in a message box.

```
Set MtbSheets = MtbProject.Worksheets
MsgBox "Number of worksheets in collection: " _
    & MtbSheets.Count
```

Worksheets Collection method - Add

Use to add *Count* `Worksheet` objects to the `Worksheets` collection.

Syntax

```
Add(Quantity Long String)
```

Arguments

Quantity
Optional. Number of worksheets to add. The default is 1.

Returns

```
Worksheet
```

Remarks

The first worksheet added is returned.

Examples

Retrieve the Worksheets collection, add two worksheets to it, then name the first one "Growth."

```
Set MtbSheets = MtbProject.Worksheets
MtbSheets.Add(2).Name = "Growth"
```

Add one worksheet to the Worksheets collection.

```
MtbSheets.Add
```

Worksheets Collection method - Delete

Use to remove all `Worksheet` objects from the `Worksheets` collection.

Syntax

```
Delete()
```

Arguments

Filename
Required. The path...

Returns

HRESULT

Remarks

To remove a single worksheet, use Remove on page 42 or the Worksheet on page 43 object method, Delete on page 48.

Example

Delete the Worksheets collection, including all its worksheets:

```
MtbSheets.Delete
```

Worksheets Collection method - Item

Use to return a Worksheet object within the Worksheets collection.

Syntax

```
Item(Index as Variant)
```

Arguments

Index
Required. The index of the worksheet as an integer (Long) from 1 - the number of worksheets in the collection, or the name on page 47 (String) of the worksheet.

Returns

Worksheet

Remarks

Because Item is the default method for the Worksheets collection, both of the following are acceptable:

```
.Worksheets.Item(2)
.Worksheets(2)
```

Examples

Retrieve the second worksheet in the Worksheets collection, name the worksheet "First Year," then print the name in a message box.

```
Set MtbSheet = MtbSheets.Item(2)
MtbSheet.Name = "First Year"
```

```
MsgBox "The second worksheet is " & _
    MtbSheet.Name
```

Note You can also retrieve the second worksheet using this command:

```
Set MtbSheet = Mtbsheets (2)
```

Retrieve the worksheet called "Second Year" and print the name in a message box.

```
Set MtbSheet = MtbSheets.Item("Second Year")
MsgBox "The current worksheet is: " & _
    MtbSheet.Name
```

Worksheets Collection method - Open

Use to open an existing Minitab worksheet disk file, read it into a `Worksheet` object, and add the `Worksheet` object to the end of the `Worksheets` collection.

Syntax

```
Open(Filename as String)
```

Arguments

Filename

Optional. The path and name of the worksheet file to be opened. If a path is not specified, the `DefaultFilePath` on page 24 is used.

Returns

```
HRESULT
```

Remarks

When you open a worksheet file, the `Name`, `Path`, and `FullName` properties of the `Worksheet` object are automatically updated. If you don't specify an extension, .MTW is automatically added.

Example

Open the Minitab worksheet "Market.mtw", retrieve the first worksheet, then print the name in a message box.

```
MtbSheets.Open ("C:\sheets\Market.mtw")
Set MtbSheet = MtbSheets(1)
MsgBox "Worksheet name: " & MtbSheet.Name
```

Worksheets Collection method - Remove

Use to delete a `Worksheet` object and remove it from the `Worksheets` collection.

Syntax

```
Remove(Index as Variant)
```

Arguments

Index

Required. The index of the worksheet as an integer (`Long`) from 1 - the number of worksheets in the collection, or the `name` on page 47 (`String`) of the worksheet.

Returns

`HRESULT`

Remarks

The same results can be achieved using the `Delete` on page 48 method of the `Worksheet` on page 43 object. To remove all worksheets, use the `Delete` on page 40 method of the `Worksheets` on page 39 collection object.

Example

Remove the first worksheet and the worksheet named "First Year" from the `Worksheets` collection.

```
MtbSheets.Remove 1
MtbSheets.Remove "First Year"
```

Worksheet object

The Worksheet object contains all the information related to an individual worksheet, including the `Columns`, `Constants`, and `Matrices` collections, which provide access to all the columns, constants, and matrices in the worksheet.

Properties

Property	Description
Columns on page 44	Returns the `Columns` collection of the worksheet.
Comment on page 44	Comment for the `Worksheet` object, saved as part of the Minitab worksheet description.
Constants on page 45	Returns the `Constants` collection of the worksheet.
Creator on page 45	Creator of the `Worksheet` object, saved as part of the Minitab worksheet description.
Date on page 46	Date of the `Worksheet` object description, saved as part of the Minitab worksheet description.
FullName on page 46	Full name of the `Worksheet` object disk file, including the path name and/or drive name, set when a `Worksheet` object is opened or saved.
Matrices on page 47	Returns the `Matrices` collection of the worksheet.
Name on page 47	Name of the `Worksheet` object and its disk file.
Path on page 47	Path of the `Worksheet` object disk file, set when a `Worksheet` object is opened or saved.

Methods

Method	Description
Delete on page 48	Use to delete a `Worksheet` object and remove it from the `Worksheets` collection.
Save on page 48	Use to save a `Worksheet` object to disk with the file name specified in the `FullName` property.
SaveAs on page 49	Use to save a `Worksheet` object to disk with the file name specified in the `Filename` argument.

Example

Retrieve the `Worksheet` object named "First Year", set the creator, date, and comment for the worksheet, then save the worksheet as "Year1."

```
Dim MtbSheet As Mtb.Worksheet
Set MtbSheet = MtbSheets("First Year")
MtbSheet.Creator = "M. Smith"
MtbSheet.Date = #6/4/2002#
MtbSheet.Comment = "1999 is the first year"
MtbSheet.SaveAs "Year1"
```

Worksheet property - Columns

Description

Returns the `Columns` collection of the worksheet.

Type

Columns on page 50

Range

N/A

Access

Read-only

Example

Retrieve the `Columns` collection.

```
Set MtbColumns = MtbSheet.Columns
```

Note This example assumes that the `MtbSheet` object was previously initialized to a valid `Worksheet` object as demonstrated in the `Worksheet` on page 43 object example.

Worksheet property - Comment

Description

Comment for the `Worksheet` object, saved as part of the Minitab worksheet description.

Type

String

Range
Valid string

Access
Read/Write

Example

Retrieve the first worksheet in the `Worksheets` collection, add a comment to the worksheet, then print the comment in a message box.

```
Set MtbSheet = MtbSheets(1)
MtbSheet.Comment = "This worksheet has the _
    old data. It needs to be updated by the _
    end of the year"
```

Worksheet property - Constants

Description
Returns the `Constants` collection of the worksheet.

Type
Constants on page 63

Range
N/A

Access
Read-only

Example

Set `MtbConstants` to the `Constants` collection of worksheet `MtbSheet`.

```
Set MtbConstants = MtbSheet.Constants
```

Note This example assumes that the `MtbSheet` object was previously initialized to a valid `Worksheet` object as demonstrated in the `Worksheet` on page 43 object example.

Worksheet property - Creator

Description
Creator of the `Worksheet` object, saved as part of the Minitab worksheet description.

Type
String

Range
Valid string

Access
Read/Write

Example

Retrieve the first worksheet in the `Worksheets` collection, specify the creator of the worksheet (M. Smith), then print the creator in a message box.

```
Set MtbSheet = MtbSheets(1)
MtbSheet.Creator = "M. Smith"
MsgBox "The creator is " & MtbSheet.Creator
```

Worksheet property - Date

Description

Date of the `Worksheet` object description, saved as part of the Minitab worksheet description.

Type

String

Range

Valid string

Access

Read/Write

Example

Retrieve the first worksheet in the `Worksheets` collection, specify the date of the worksheet, then print the date in a message box.

```
Set MtbSheet = MtbSheets(1)
MtbSheet.Date = #6/4/2002#
MsgBox "The date is " & MtbSheet.Date
```

Worksheet property - FullName

Description

Full name of the `Worksheet` object disk file, including the path name and/or drive name, set when a `Worksheet` object is opened or saved.

Type

String

Range

Valid file name, including path name and/or drive name

Access

Read/Write

Example

Display the `FullName` property (path and name) of the worksheet in a message box.

```
MsgBox "The FullName is " & _
    MtbSheet.FullName
```

Worksheet property - Matrices

Description

Returns the `Matrices` collection of the worksheet.

Type

Matrices on page 72

Range

N/A

Access

Read-only

Example

Set `MtbMatrices` to the `Constants` collection of worksheet `MtbSheet`.

```
Set MtbMatrices = MtbSheet.Matrices
```

Note This example assumes that the `MtbSheet` object was previously initialized to a valid `Worksheet` object as demonstrated in the `Worksheet` on page 43 object example.

Worksheet property - Name

Description

Name of the `Worksheet` object and its disk file. It is also the name of the file if the `Worksheet` object is saved to disk. Setting `Name` automatically updates the file name portion of the `FullName` property.

Type

String

Range

Any valid file name

Access

Read/Write

Do not include the path when setting `Name`.

Example

Retrieve the second worksheet in the `Worksheets` collection, name the worksheet "Second Year," then print the name in a message box.

```
Set MtbSheet = MtbSheets(2)
MtbSheet.Name = "Second Year"
MsgBox "The second worksheet is " & _
    MtbSheet.Name
```

Worksheet property - Path

Description

Path of the `Worksheet` object disk file, set when a `Worksheet` object is opened or saved.

Type
String

Range
Valid path name. It may include the drive name.

Access
Read-only

Example

Display the `Path` property of the worksheet in a message box.

```
MsgBox "The Path is " & _
    MtbSheet.Path
```

Worksheet method - Delete

Use to delete a `Worksheet` object and remove it from the `Worksheets` collection.

Syntax

```
Delete()
```

Returns

```
HRESULT
```

Remarks

The same results can be achieved using the `Remove` on page 42 method of the `Worksheets` on page 39 collection object. To delete all worksheets, use the `Delete` on page 40 method of the `Worksheets` on page 39 collection object.

Example

Delete the `Worksheet` object from the `Worksheets` collection.

```
 MtbSheet.Delete
```

Worksheet method - Save

Use to save a `Worksheet` object to disk with the file name specified in the `FullName` property.

Syntax

```
Save(Filename as String)
```

Returns

```
HRESULT
```

Remarks

When you save a worksheet, the `Name`, `Path`, and `FullName` properties of the `Worksheet` object are automatically updated. If you don't specify an extension, .MTW is automatically added to the worksheet name. If the `FullName` property is null, then the worksheet is saved to Minitab.MTW at the default file path.

Example

Save the current worksheet.

```
MtbSheet.Save
```

Worksheet method - SaveAs

Use to save a `Worksheet` object to disk with the file name specified in the `Filename` argument.

Syntax

```
SaveAs(Filename as String, Replace as Boolean, Version as Long)
```

Arguments

Filename
Optional. Path and file name to use when saving the file. If a path is not specified, then the `DefaultFilePath` on page 24 is used.

Replace
Optional. If `True`, an existing file with the same name will be overwritten. The default is `False`.

Version
Optional. The Minitab version number to save the worksheet as. If not specified, the current version number is used.

Returns

`HRESULT`

Remarks

When you save a worksheet, the `Name`, `Path`, and `FullName` properties of the `Worksheet` object are automatically updated. If you don't specify an extension, .MTW is automatically added to the worksheet name.

Examples

Save the current worksheet as "April Totals" at the file path C:\MTBsheets.

```
MtbSheet.SaveAs "C:\MTBsheets\April Totals"
```

Save the current worksheet as "April Totals" at the default file path, overwriting the existing "April Totals".

```
MtbSheet.SaveAs "April Totals", True
```

Columns Collection object

The `Columns` collection is a set of all the `Column` objects within a Worksheet object. It supports the standard collection properties and methods.

The `Columns` collection for a worksheet is empty by default.

Properties

Property	Description
Count on page 50	Number of `Column` objects within the `Columns` collection.

Methods

Method	Description
Add on page 51	Use to add *`Quantity`* `Column` objects to the `Columns` collection in the position before *`Before`* or after *`After`*.
Delete on page 52	Use to remove all `Column` objects from the `Columns` collection.
Item on page 52	Use to return a `Column` object within the `Columns` collection.
Remove on page 53	Use to delete a `Column` object and remove it from the `Columns` collection.

Example

Retrieve the `Columns` collection (`MtbColumns`), add two columns to the end of the collection and name the first one "Sales," then remove the second column from the `Columns` collection.

```
Dim MtbApp As Mtb.Application
Dim MtbSheet As Mtb.Worksheet
Dim MtbColumns As Mtb.Columns

Set MtbApp = New Mtb.Application
Set MtbSheet = MtbApp.ActiveProject.ActiveWorksheet
Set MtbColumns = MtbSheet.Columns

MtbColumns.Add(, , 2).Name = "Sales"
MtbColumns.Remove 2
```

Columns Collection property - Count

Description
Number of `Column` objects within the `Columns` collection.

Type
Long

Range
0 - number of `Column` objects within the `Columns` collection

Access
Read-only

Example

Retrieve the Columns collection, then display in a message box the number of Column objects in the Columns collection.

```
Dim MtbApp As Mtb.Application
Dim MtbSheet As Mtb.Worksheet
Dim MtbColumns As Mtb.Columns

Set MtbApp = New Mtb.Application
Set MtbSheet = MtbApp.ActiveProject.ActiveWorksheet
Set MtbColumns = MtbSheet.Columns

MsgBox "Number of columns in collection: " & _
    MtbColumns.Count
```

Columns Collection method - Add

Use to add *Quantity* Column objects to the Columns collection in the position before *Before* or after *After*.

Syntax

```
Add(Before as Variant, After as Variant, Quantity as Long)
```

Arguments

Before
Optional. Column object to add new columns before.

After
Optional. Column object to add new columns after.

Quantity
Optional. Number of columns to add. The default is 1.

Returns

Column on page 54

Remarks

You can specify either *Before* or *After*, but not both. Use an integer (Long) from 1 - the number of columns in the collection, or the name on page 58 (String) of a column. If neither *Before* nor *After* is specified, then the columns are added after the last column in the collection.

The first column added is returned.

Examples

Create a Minitab Application object, add four columns to the active worksheet, and name the first column "Year."

```
Dim MtbApp As Mtb.Application
Dim MtbSheet As Mtb.Worksheet
Dim MtbColumns As Mtb.Columns
Dim MtbColumn As Mtb.Column

Set MtbApp = New Mtb.Application
```

```
Set MtbSheet = MtbApp.ActiveProject.ActiveWorksheet
Set MtbColumns = MtbSheet.Columns

MtbColumns.Add(, , 4).Name = "Year"
```

Add one column to it after the last column.

```
MtbColumns.Add()
```

Add two columns to the `Columns` collection before column three, and name the first column "First Time."

```
MtbColumns.Add(3,,2).Name = "First Time"
```

Add two columns to the `Columns` collection after column three.

```
MtbColumns.Add(,3,2)
```

Add four columns to the `Columns` collection before the "Year" column and name the first column "Next Year."

```
MtbColumns.Add("Year",,4).Name = "Next Year"
```

Add two columns to the `Columns` collection after the "Year" column.

```
MtbColumns.Add(,"Year",2)
```

Columns Collection method - Delete

Use to remove all `Column` objects from the `Columns` collection.

Syntax

```
Delete()
```

Returns

```
HRESULT
```

Remarks

To remove a single column, use `Remove` on page 53 or the `Column` object method, `Delete` on page 60.

Example

Delete the `Columns` collection (`MtbColumns`), including all its columns.

```
Dim MtbApp As Mtb.Application
Dim MtbSheet As Mtb.Worksheet
Dim MtbColumns As Mtb.Columns

Set MtbApp = New Mtb.Application
Set MtbSheet = MtbApp.ActiveProject.ActiveWorksheet
Set MtbColumns = MtbSheet.Columns

MtbColumns.Delete
```

Columns Collection method - Item

Use to return a `Column` object within the `Columns` collection.

Syntax

```
Item(Index as Variant)
```

Arguments

Index

Required. The index of the column as an integer (`Long`) from 1 - the number of columns in the collection, or the `name` on page 47 (`String`) of the column.

Returns

`Column` on page 54

Remarks

Because `Item` is the default method for the `Columns` collection, both of the following are acceptable:

```
.Columns.Item(2)
.Columns(2)
```

Examples

Create a Minitab `Application` object, add four columns to the active worksheet, and name the first column "Mileage."

```
Dim MtbApp As Mtb.Application
Dim MtbSheet As Mtb.Worksheet
Dim MtbColumns As Mtb.Columns
Dim MtbColumn As Mtb.Column

Set MtbApp = New Mtb.Application
Set MtbSheet = MtbApp.ActiveProject.ActiveWorksheet
Set MtbColumns = MtbSheet.Columns

MtbColumns.Add(, , 4).Name = "Mileage"
```

Retrieve the second column in the `Columns` collection, name the column "Range," then print the name in a message box.

```
Set MtbColumn = MtbColumns.Item(2)
MtbColumn.Name = "Range"
MsgBox "The second column is " & _
    MtbColumn.Name
```

You can also retrieve the second column using this command:

```
Set MtbColumn = MtbColumns(2)
```

Retrieve the column called "Mileage" then print the name in a message box.

```
Set MtbColumn = MtbColumns.Item("Mileage")
MsgBox "The current column is " & _
    MtbColumn.Name
```

Columns Collection method - Remove

Use to delete a `Column` object and remove it from the `Columns` collection.

Syntax

```
Remove(Index as Variant)
```

Arguments

Index

Required. The index of the column as an integer (`Long`) from 1 - the number of columns in the collection, or the name on page 58 (`String`) of the column.

Returns

`HRESULT`

Remarks

The same results can be achieved using the `Delete` on page 60 method of the `Column` on page 54 object. To remove all columns, use the `Delete` on page 52 method of the `Columns` on page 50 collection object.

Examples

Create a Minitab `Application` object, add four columns to the active worksheet, and name the first column "First Year."

```
Dim MtbApp As Mtb.Application
Dim MtbSheet As Mtb.Worksheet
Dim MtbColumns As Mtb.Columns
Dim MtbColumn As Mtb.Column

Set MtbApp = New Mtb.Application
Set MtbSheet = MtbApp.ActiveProject.ActiveWorksheet
Set MtbColumns = MtbSheet.Columns

MtbColumns.Add(, , 4).Name = "First Year"
```

Remove the second column and the column named "First Year" from the `Columns` collection:

```
MtbColumns.Remove 2
MtbColumns.Remove "First Year"
```

Column object

The `Column` object contains all the information related to an individual column. The data type on page 7 for each `Column` object can be `Text`, `Numeric`, `DateTime`, or `DataUnassigned`.

Properties

Property	Description
Comment on page 56	Description of the Column object.
DataType on page 56	Type of data in the `Column` object.
Formula on page 56	Formula for the `Column` object.
FormulaStatus on page 57	Status of the `Formula` property for the `Column` object.
MissingCount on page 58	Number of missing data rows in the `Column` object
Name on page 58	Name of the `Column` object.
Number on page 59	Number of the `Column` object within the `Columns` collection.

Property	Description
RowCount on page 59	Number of rows in the `Column` object
ValueOrderType on page 59	Order in which values from text columns will be displayed in output.

Methods

Method	Description
Clear on page 59	Use to clear the data in the `Column` object without deleting the `Column` object from the `Columns` collection.
Delete on page 60	Use to delete a `Column` object and remove it from the `Columns` collection.
GetData on page 61	Use to get *`NumRows`* of data from a `Column` object, starting at *`StartRow`*.
SetData on page 61	Use to set *`NumRows`* of *`Data`* in the `Column` object, beginning at *`StartRow`*.
SetValueOrder on page 62	Use to set the order in which text values are displayed in output. Column must be of type `Text` on page 7 or `DataUnassigned` on page 7.

Examples

Create a Minitab `Application` object and add a `Column` object to the `Columns` collection of the active worksheet. Define and populate the array "`arrSales`" with the column information, retrieve the `Column` object (`MtbColumn`), name it "Sales," and place the information in `arrSales` into the "Sales" column. Finally, set the `Comment` property of the new column.

```
Dim MtbApp As Mtb.Application
Dim MtbColumns As Mtb.Columns
Dim MtbColumn As Mtb.Column
Dim arrSales(1 To 8) As Double
    arrSales(1) = 94
    arrSales(2) = 99
    arrSales(4) = 92
    arrSales(5) = 106
    arrSales(6) = 116
    arrSales(7) = 113
    arrSales(8) = 108

Set MtbApp = New Mtb.Application
Set MtbColumns = MtbApp.ActiveProject.ActiveWorksheet.Columns
Set MtbColumn = MtbColumns.Add(, , 1)

MtbColumn.Name = "Sales"
MtbColumn.SetData arrSales, 1, 8

MtbColumn.Comment = "Sales data for 1999"
```

Create a Minitab `Application` object and add two `Column` objects to the `Columns` collection of the active worksheet. Retrieve the second column, add the value "1993" to the third row, set the `Name` property to "Second Year," set the `ValueOrderType` property to 0, and set the `Comment` property (column description) to "New column for second year data." Finally, display message boxes with the values of the column's `Name`, `Number`, `RowCount`, `MissingCount`, `DataType`, `ValueOrderType`, and `Comment` properties.

```
Dim MtbApp As New Mtb.Application
Dim MtbColumns As Mtb.Columns
Dim MtbColumn As Mtb.Column

Set MtbColumns = MtbApp.ActiveProject.ActiveWorksheet.Columns
MtbColumns.Add , , 2
Set MtbColumn = MtbColumns(2)
MtbColumn.SetData "1993", 3
```

```
'Set property values
MtbColumn.Name = "Second Year"
MtbColumn.SetValueOrder = 0
MtbColumn.Comment = "New column for second year data."

'Display messages with column property values
MsgBox "The second column is " & _
    MtbColumn.Name
MsgBox "This is column number: " & _
    MtbColumn.Number
MsgBox "This column has this many rows: " & _
    MtbColumn.RowCount
MsgBox "This column has this many missing rows: " & _
    MtbColumn.MissingCount
MsgBox "The data type of the column is " & _
    MtbColumn.DataType
MsgBox "The ValueOrderType is " & _
    MtbColumn.ValueOrderType
MsgBox "The column description is " & _
    MtbColumn.Comment
```

Column property - Comment

Description

Description of the Column object.

Type

String

Range

Valid string

Access

Read/Write

Column property - DataType

Description

Type of data in the `Column` object.

Type

MtbDataTypes on page 7

Range

Any `MtbDataTypes` constant

Access

Read-only

Column property - Formula

Description

Formula for the `Column` object.

Type
String

Range
Valid string

Access
Read-only

Example

Create 30 rows of random data in column C1, then create a formula that sets the value of C2 equal to the square of C1. Display a message box showing the value of the `Formula` and `FormulaStatus` properties for C2. Finally, change to manual formula calculation using the `CFMANUALLY` session command, change a value in C1, and display the same message. Notice that `FormulaStatus` changes from 1 to 2.

```
Dim MtbApp As New Mtb.Application
Dim MtbSheet As Mtb.Worksheet
Dim MtbCol1, MtbCol2 As Mtb.Column
With MtbApp
    .UserInterface.Visible = True
    .ActiveProject.ExecuteCommand ("Rand 30 C1")
    .ActiveProject.ExecuteCommand ("Formula C2=C1**2")
    Set MtbSheet = .ActiveProject.Worksheets(1)
    With MtbSheet
        Set MtbCol1 = .Columns(1)
        Set MtbCol2 = .Columns(2)
        MsgBox ( _
            "Column 2 formula: " & MtbCol2.Formula & vbCrLf & _
            "Column 2 formula status: " & MtbCol2.FormulaStatus)
    End With
    .ActiveProject.ExecuteCommand ("CFMANUALLY")
    MtbCol1.SetData 20, 3, 1
    MsgBox ( _
        "Column 2 formula: " & MtbCol2.Formula & vbCrLf & _
        "Column 2 formula status: " & MtbCol2.FormulaStatus)
End With
```

Column property - FormulaStatus

Description
Status of the `Formula` property for the `Column` object.

Type
MtbFormulaStatusTypes on page 7

Range
Any `MtbFormulaStatusTypes` constant

Access
Read-only

Example

Create 30 rows of random data in column C1, then create a formula that sets the value of C2 equal to the square of C1. Display a message box showing the value of the `Formula` and `FormulaStatus` properties for C2. Finally, change

to manual formula calculation using the `CFMANUALLY` session command, change a value in C1, and display the same message. Notice that `FormulaStatus` changes from 1 to 2.

```
Dim MtbApp As New Mtb.Application
Dim MtbSheet As Mtb.Worksheet
Dim MtbCol1, MtbCol2 As Mtb.Column
With MtbApp
    .UserInterface.Visible = True
    .ActiveProject.ExecuteCommand ("Rand 30 C1")
    .ActiveProject.ExecuteCommand ("Formula C2=C1**2")
    Set MtbSheet = .ActiveProject.Worksheets(1)
    With MtbSheet
        Set MtbCol1 = .Columns(1)
        Set MtbCol2 = .Columns(2)
        MsgBox ( _
            "Column 2 formula: " & MtbCol2.Formula & vbCrLf & _
            "Column 2 formula status: " & MtbCol2.FormulaStatus)
    End With
    .ActiveProject.ExecuteCommand ("CFMANUALLY")
    MtbCol1.SetData 20, 3, 1
    MsgBox ( _
        "Column 2 formula: " & MtbCol2.Formula & vbCrLf & _
        "Column 2 formula status: " & MtbCol2.FormulaStatus)
End With
```

Column property - MissingCount

Description

Number of missing data rows in the `Column` object

Type

Long

Range

N/A

Access

Read-only

Column property - Name

Description

Name of the `Column` object.

Type

String

Range

Valid string

Access

Read/Write

Column property - Number

Description
Number of the `Column` object within the `Columns` collection.

Type
Long

Range
1 - number of `Column` objects within the `Columns` collection (current Minitab limit is 4000)

Access
Read-only

Column property - RowCount

Description
Number of rows in the `Column` object

Type
Long

Range
N/A

Access
Read-only

Column property - ValueOrderType

Description
Order in which values from text columns will be displayed in output.

Type
MtbValueOrderTypes on page 8

Range
Any `MtbValueOrderTypes` constant

Access
Read-only

Column method - Clear

Use to clear the data in the `Column` object without deleting the `Column` object from the `Columns` collection.

Syntax

```
Clear()
```

Returns

HRESULT

Examples

Create a Minitab Application object and add two columns to the active worksheet. Retrieve the second column in the Columns collection, name the column "Second Year," and the value "1993" to the third row.

```
Dim MtbApp As Mtb.Application
Dim MtbColumns As Mtb.Columns
Dim MtbColumn As Mtb.Column

Set MtbApp = New Mtb.Application
Set MtbColumns = MtbApp.ActiveProject.ActiveWorksheet.Columns
MtbColumns.Add , , 2

Set MtbColumn = MtbColumns(2)
MtbColumn.Name = "Second Year"
MtbColumn.SetData "1993", 3
```

Clear the data from the Column object without deleting the column itself.

```
MtbColumn.Clear
```

Column method - Delete

Use to delete a Column object and remove it from the Columns collection.

Syntax

```
Delete()
```

Returns

HRESULT

Remarks

The same results can be achieved using the Remove on page 53 method of the Columns on page 50 collection object. To delete all columns, use the Delete on page 52 method of the Columns on page 50 collection object.

Example

Create a Minitab Application object and add two columns to the active worksheet. Retrieve the second column in the Columns collection, name the column "Second Year," and add the value "1993" to the third row. Finally, delete the Column object from the Columns collection.

```
Dim MtbApp As Mtb.Application
Dim MtbColumns As Mtb.Columns
Dim MtbColumn As Mtb.Column

Set MtbApp = New Mtb.Application
Set MtbColumns = MtbApp.ActiveProject.ActiveWorksheet.Columns
MtbColumns.Add , , 2

Set MtbColumn = MtbColumns(2)
MtbColumn.Name = "Second Year"
MtbColumn.SetData "1993", 3

MtbColumn.Delete
```

Column method - GetData

Use to get *NumRows* of data from a Column object, starting at *StartRow*.

Syntax

```
GetData(StartRow as Long, NumRows as Long)
```

Arguments

StartRow
Optional. First row to get. The default is 1.

NumRows
Optional. Number of rows to get. The default is 1.

Returns

```
Variant
```

Remarks

If neither *StartRow* nor *NumRows* is specified, then GetData gets all rows.

Examples

Use the GetData method to populate a variant (array) with all the values from the current column. Use a loop to print the values in the Immediate window.

```
Dim i As Integer
Dim cvQ1sales As Variant
cvQ1sales = Mtbcolumn.GetData()

For i = 0 To 7
    Debug.Print cvQ1sales(i)
Next i
```

Get one value (the first value) from the column and print it in a message box.

```
MsgBox "The value is " & MtbColumn.GetData(,1)
```

Get one value (the second value) from the column and print it in a message box.

```
MsgBox "The value is " & MtbColumn.GetData(2)
```

Column method - SetData

Use to set *NumRows* of *Data* in the Column object, beginning at *StartRow*.

Syntax

```
SetData(Data as Variant, StartRow as Long, NumRows as Long)
```

Arguments

Data
Data to write to the column. Can be numeric, text, or date/time.

StartRow
Optional. First row to set. The default is 1.

NumRows
Optional. Number of rows to set. The default is 1.

Returns

`HRESULT`

Remarks

If neither *StartRow* nor *NumRows* is specified, `SetData` sets the entire column and deletes all previous entries. Otherwise, entries outside the specified range of rows are not affected.

Examples

Retrieve the first column in the `Columns` collection then populate rows 1-8 with the contents of the array `arrIndex`.

```
Set MtbColumn = MtbColumns(1)
MtbColumn.SetData arrIndex, 1, 8
```

Place the value 10 in row 20 of the current Minitab column, then print the value for row 20 in the Immediate window.

```
MtbColumn.SetData 10, 20
Debug.Print MtbColumn.GetData(20)
```

Place "Green" in the first row of the column and print it in a Message Box.

```
MtbColumn.SetData "Green"
MsgBox "The value is " & MtbColumn.GetData(1)
```

Column method - SetValueOrder

Use to set the order in which text values are displayed in output. Column must be of type `Text` on page 7 or `DataUnassigned` on page 7.

Syntax

```
SetValueOrder(ValueOrderType as MtbValueOrderTypes, UserDefinedOrder as Variant)
```

Arguments

ValueOrderType
Required. Value order type for column. May be any MtbValueOrderTypes on page 8 constant.

UserDefinedOrder
Optional. Variant array specifying user defined value order. Required for MtbValueOrderTypes = 2, ignored otherwise.

Returns

HRESULT

Example

Create a new Minitab Application object, create and add text data to column 1, then set a user-defined value order for the column.

```
Dim App As New Mtb.Application
Dim arData(5), arOrder(2)

arData(0) = "a"
arData(1) = "a"
arData(2) = "b"
arData(3) = "b"
arData(4) = "c"
arData(5) = "c"

arOrder(0) = "c"
arOrder(1) = "a"
arOrder(2) = "b"

Set Project = App.ActiveProject
Project.ActiveWorksheet.Columns.Add , , 3
Project.ActiveWorksheet.Columns(1).SetData arData
Project.ActiveWorksheet.Columns(1).SetValueOrder 2, arOrder
```

Constants Collection object

The Constants collection is a set of all the Constant objects within a Worksheet object. It supports the standard collection properties and methods.

The Constants collection for a worksheet is empty by default.

Properties

Property	Description
Count on page 64	Number of Constant objects within the Constants collection.

Methods

Method	Description
Add on page 64	Use to add *Count* Constant objects to the Constants collection in the position before *Before* or after *After*.
Delete on page 65	Use to remove all Constant objects from the Constants collection.
Item on page 65	Use to return a Constant object within the Constants collection.
Remove on page 66	Use to delete a Constant object and remove it from the Constants collection.

Example

Retrieve the Constants collection and add a constant, then name it "SalesFactor."

```
Dim MtbConstants As Mtb.Constants
Set MtbConstants = MtbSheet.Constants
MtbConstants.Add.Name = "SalesFactor"
```

Constants Collection property - Count

Description

Number of Constant objects within the Constants collection.

Type

Long

Range

0 - number of Constant objects in the Constants collection

Access

Read-only

Example

Retrieve the Constants collection, add four Constant objects to it, then display the number of Constant objects in the Constants collection in a message box.

```
Set MtbConstants = MtbSheet.Constants
MtbConstants.Add (4)
MsgBox "Number of constants in collection: " _
    & MtbConstants.Count
```

Constants Collection method - Add

Use to add *Count* Constant objects to the Constants collection in the position before *Before* or after *After*.

Syntax

```
Add(Before as Variant, After as Variant, Quantity as Long)
```

Arguments

Before

Optional. Constant object to add new constants before.

After

Optional. Constant object to add new constants after.

Quantity

Optional. Number of constants to add. The default is 1.

Returns

Constant on page 67

Remarks

You can specify either *Before* or *After*, but not both. Use an integer (Long) from 1 - the number of constants in the collection, or the name on page 70 (String) of a constant. If neither *Before* nor *After* is specified, then the constants are added after the last constant in the collection.

The first constant added is returned.

Examples

Retrieve the Constants collection and add one constant to it after the last constant.

```
Set MtbConstants = MtbSheet.Constants
MtbConstants.Add
```

Add two constants to the Constants collection before the third constant, then name the first constant "Factor1."

```
MtbConstants.Add(3,,2).Name = "Factor1"
```

Add two constants to the Constants collection after the third constant.

```
MtbConstants.Add(,3,2)
```

Add four constants to the Constants collection before the "Factor1" constant, then name the first constant "NewFactor1."

```
MtbConstants.Add("Factor1",,4).Name = "NewFactor1"
```

Add two constants to the Constants collection after the "Factor1" constant.

```
MtbConstants.Add(,"Factor1",2)
```

Constants Collection method - Delete

Use to remove all Constant objects from the Constants collection.

Syntax

```
Delete()
```

Returns

```
HRESULT
```

Remarks

To remove a single constant, use Remove on page 66 or the Delete on page 71 method of the Constant on page 67 object.

Example

Delete the Constants collection, including all its constants.

```
MtbConstants.Delete
```

Constants Collection method - Item

Use to return a Constant object within the Constants collection.

Syntax

```
Item(Index as Variant)
```

Arguments

Index

Required. The index of the constant as an integer (`Long`) from 1 - the number of constants in the collection, or the name on page 70 (`String`) of the constant.

Returns

`Constant` on page 67

Remarks

Because `Item` is the default method for the `Constants` collection, both of the following are acceptable:

```
.Constants.Item(2)
.Constants(2)
```

Examples

Retrieve the second constant in the `Constants` collection, name the constant "Conversion Factor", and print the name in a message box.

```
Set MtbConstant = MtbConstants.Item(2)
MtbConstant.Name = "Conversion Factor"
MsgBox "The second constant is " & _
    MtbConstant.Name
```

Retrieve the second constant using this command.

```
Set MtbConstant = Mtbconstants (2)
```

Retrieve the constant called "Metric" and print the name in a message box.

```
Set MtbConstant = MtbConstants.Item("Metric")
MsgBox "The current constant is " & _
    MtbConstant.Name
```

Constants Collection method - Remove

Use to delete a `Constant` object and remove it from the `Constants` collection.

Syntax

```
Remove(Index as Variant)
```

Arguments

Index

Required. The index of the constant as an integer (`Long`) from 1 - the number of constants in the collection, or the name on page 70 (`String`) of the constant.

Returns

HRESULT

Remarks

The same results can be achieved using the Delete on page 71 method of the Constant on page 67 object. To remove all constants, use the Delete on page 65 method of the Constants on page 63 collection object.

Example

Remove the first constant and the constant named "Factor1" from the Constants collection.

```
MtbConstants.Remove 1
MtbConstants.Remove "Factor1"
```

Constant object

The Constant object contains all the information related to an individual constant. The Constant object can contain numeric or text values.

Properties

Property	Description
Comment on page 68	Description of the Constant object.
DataType on page 68	Type of the data in the Constant object.
Formula on page 68	Formula for the Constant object.
FormulaStatus on page 69	Status of the Formula property for the Constant object.
Name on page 70	Name of the Constant object
Number on page 70	Number of the Constant object.

Methods

Method	Description
Delete on page 71	Use to delete a Constant object and remove it from the Constants collection.
GetData on page 71	Use to return the value stored in the constant.
SetData on page 72	Use to set the value of the Constant object using the value in *Data*.

Example

Define the Constant object (MtbConstant) and retrieve the first constant in the Constants collection, set the value of the constant to 22.2, then use the GetData method to print the constant in the Immediate window.

```
Dim MtbConstant As Mtb.Constant
Set MtbConstant = MtbConstants(1)
MtbConstant.SetData 22.2
Debug.Print MtbConstant.GetData
```

Constant property - Comment

Description
Description of the `Constant` object.

Type
String

Range
Valid string

Access
Read/Write

Example

Add a comment to the constant and print the comment in a message box.

```
MtbConstant.Comment = "This constant converts _
    English to metric."
MsgBox MtbConstant.Comment
```

Constant property - DataType

Description
Type of the data in the `Constant` object.

Type
MtbDataTypes on page 7

Range
Any `MtbDataTypes` constant except `DateTime`

Access
Read-only

Example

Display the data type of the constant in a message box.

```
MsgBox "The data type of the constant is " & _
    Mtbconstant.DataType
```

Constant property - Formula

Description
Formula for the `Constant` object.

Type
String

Range
Valid string

Access
Read-only

Example

Create two constants, K1 and K2. Set the value of K1 to 3 and create a formula that makes the value of K2 equal to K1 squared. Display a message box showing the values of both constants as well as the `Formula` and `FormulaStatus` properties of K2. Finally, change to manual formula calculation using the `CFMANUALLY` session command, change the value of K1, then display the same message. Notice that `FormulaStatus` changes from 1 to 2.

```
Dim MtbApp As New Mtb.Application
Dim MtbSheet As Mtb.Worksheet
Dim MtbConst1, MtbConst2 As Mtb.Constant
With MtbApp
    .UserInterface.Visible = True
    With .ActiveProject.Worksheets(1)
        Set MtbConst1 = .Constants.Add
        Set MtbConst2 = .Constants.Add
        MtbConst1.SetData (3)
    End With
    .ActiveProject.ExecuteCommand ("FORMULA K2 = K1**2")
    MsgBox ( _
        "K1 = " & MtbConst1.GetData & vbCrLf & _
        "K2 formula = " & MtbConst2.Formula & vbCrLf & _
        "K2 = " & MtbConst2.GetData & vbCrLf & _
        "K2 formula status = " & MtbConst2.FormulaStatus)
    .ActiveProject.ExecuteCommand ("CFMANUALLY")
    MtbConst1.SetData (5)
    MsgBox ( _
        "K1 = " & MtbConst1.GetData & vbCrLf & _
        "K2 formula = " & MtbConst2.Formula & vbCrLf & _
        "K2 = " & MtbConst2.GetData & vbCrLf & _
        "K2 formula status = " & MtbConst2.FormulaStatus)
End With
```

Constant property - FormulaStatus

Description
Status of the `Formula` property for the `Constant` object.

Type
MtbFormulaStatusTypes on page 7

Range
Any `MtbFormulaStatusTypes` constant

Access
Read-only

Example

Create two constants, K1 and K2. Set the value of K1 to 3 and create a formula that makes the value of K2 equal to K1 squared. Display a message box showing the values of both constants as well as the `Formula` and `FormulaStatus` properties of K2. Finally, change to manual formula calculation using the `CFMANUALLY` session command, change the value of K1, then display the same message. Notice that `FormulaStatus` changes from 1 to 2.

```
Dim MtbApp As New Mtb.Application
Dim MtbSheet As Mtb.Worksheet
Dim MtbConst1, MtbConst2 As Mtb.Constant
```

```
With MtbApp
    .UserInterface.Visible = True
    With .ActiveProject.Worksheets(1)
        Set MtbConst1 = .Constants.Add
        Set MtbConst2 = .Constants.Add
        MtbConst1.SetData (3)
    End With
    .ActiveProject.ExecuteCommand ("FORMULA K2 = K1**2")
    MsgBox ( _
        "K1 = " & MtbConst1.GetData & vbCrLf & _
        "K2 formula = " & MtbConst2.Formula & vbCrLf & _
        "K2 = " & MtbConst2.GetData & vbCrLf & _
        "K2 formula status = " & MtbConst2.FormulaStatus)
    .ActiveProject.ExecuteCommand ("CFMANUALLY")
    MtbConst1.SetData (5)
    MsgBox ( _
        "K1 = " & MtbConst1.GetData & vbCrLf & _
        "K2 formula = " & MtbConst2.Formula & vbCrLf & _
        "K2 = " & MtbConst2.GetData & vbCrLf & _
        "K2 formula status = " & MtbConst2.FormulaStatus)
End With
```

Constant property - Name

Description

Name of the `Constant` object

Type

String

Range

Valid string

Access

Read/Write

Example

Retrieve the second constant in the `Constants` collection, name the constant "Factor2," and print the name in a message box.

```
Set MtbConstant = MtbConstants(2)
MtbConstant.Name = "Factor2"
MsgBox "The second constant is " & _
    MtbConstant.Name
```

Constant property - Number

Description

Number of the `Constant` object.

Type

Long

Range

1 - number of `Constant` objects in the `Constants` collection (current Minitab limit is 1000)

Access
Read-only

Example

Display in a message box the number of the `Constant` object within the `Constants` collection.

```
MsgBox "This is constant number " & _
    MtbConstant.Number
```

Constant method - Delete

Use to delete a `Constant` object and remove it from the `Constants` collection.

Syntax

```
Delete()
```

Returns

```
HRESULT
```

Remarks

The same results can be achieved using the `Remove` on page 66 method of the `Constants` on page 63 collection object. To delete all constants, use the `Delete` on page 65 method of the `Constants` on page 63 collection object.

Example

Delete the `Constant` object from the `Constants` collection.

```
MtbConstant.Delete
```

Constant method - GetData

Use to return the value stored in the constant.

Syntax

```
GetData()
```

Returns

```
Variant
```

Example

Print the value of the constant in a message box.

```
MsgBox "The constant value is " &
    MtbConstant.GetData
```

Constant method - SetData

Use to set the value of the `Constant` object using the value in *`Data`*.

Syntax

```
SetData(Data as Variant)
```

Arguments

`Data`
Required. Value to be stored in constant. Can be numeric or text.

Returns

`HRESULT`

Examples

Retrieve the first constant in the `Constants` collection, then set it equal to "Purple."

```
Set MtbConstant = MtbConstants(1)
MtbConstant.SetData "Purple"
```

Set the value of the current constant to 4.275, then print the value in the Immediate window.

```
MtbConstant.SetData 4.275
Debug.Print MtbConstant.GetData
```

Matrices Collection object

The `Matrices` collection is a set of all the `Matrix` objects within a `Worksheet` object. It supports the standard collection properties and methods.

The `Matrices` collection for a worksheet is empty by default.

Properties

Property	Description
Count on page 73	Number of `Matrix` objects within the `Matrices` collection.

Methods

Method	Description
Add on page 73	Use to add *`Count`* `Matrix` objects to the `Matrices` collection in the position before `Before` or after `After`.
Delete on page 74	Use to remove all `Matrix` objects from the `Matrices` collection.
Item on page 74	Use to return a `Matrix` object within the `Matrices` collection.
Remove on page 75	Use to delete a `Matrix` object and remove it from the `Matrices` collection.

Example

Retrieve the Matrices collection (MtbMatrices) , add a matrix to it, then name it "Weather Factors."

```
Dim MtbMatrices As Mtb.Matrices
Set MtbMatrices = MtbSheet.Matrices
MtbMatrices.Add.Name = "Weather Factors"
```

Matrices Collection property - Count

Description

Number of Matrix objects within the Matrices collection.

Type

Long

Range

0 - number of Matrix objects in the Matrices collection

Access

Read-only

Example

Retrieve the Matrices collection, add four Matrix objects to it, then display in a message box the number of Matrix objects in the Matrices collection.

```
Set MtbMatrices = MtbSheet.Matrices
MtbMatrices.Add (4)
MsgBox "Number of matrices in collection:" &
    MtbMatrices.Count
```

Matrices Collection method - Add

Use to add *Count* Matrix objects to the Matrices collection in the position before Before or after After.

Syntax

```
Add(Before as Variant, After as Variant, Quantity as Long)
```

Arguments

Before

Optional. Matrix object to add new matrices before.

After

Optional. Matrix object to add new matrices after.

Quantity

Optional. Number of matrices to add. The default is 1.

Returns

Matrix on page 76

Remarks

You can specify either `Before` or `After`, but not both. Use an integer (`Long`) from 1 - the number of matrices in the collection, or the name on page 78 (`String`) of a matrix. If neither `Before` nor `After` is specified, then the matrices are added after the last matrix in the collection.

The first matrix added is returned.

Examples

Retrieve the `Matrices` collection and add one matrix to it after the last matrix.

```
Set MtbMatrices = MtbSheet.Matrices
MtbMatrices.Add
```

Add two matrices to the `Matrices` collection before the third matrix then name the first matrix "Gradient1."

```
MtbMatrices.Add(3,,2).Name = "Gradient1"
```

Add two matrices to the `Matrices` collection after the third matrix.

```
MtbMatrices.Add(,3,2)
```

Add four matrices to the `Matrices` collection before the "Gradient1" matrix, then name the first matrix "NewGradient1."

```
MtbMatrices.Add("Gradient1",,4).Name = "NewGradient1"
```

Add two matrices to the `Matrices` collection after the "Gradient1" matrix.

```
MtbMatrices.Add(,"Gradient1",2)
```

Matrices Collection method - Delete

Use to remove all `Matrix` objects from the `Matrices` collection.

Syntax

```
Delete()
```

Returns

```
HRESULT
```

Remarks

To remove a single matrix, use `Remove` on page 75 or the `Delete` on page 79 method of the `Matrix` on page 76 object.

Example

Delete the `Matrices` collection, including all its `Matrix` objects.

```
MtbMatrices.Delete
```

Matrices Collection method - Item

Use to return a `Matrix` object within the `Matrices` collection.

Syntax

```
Item(Index as Variant)
```

Arguments

Index

Required. The index of the matrix as an integer (`Long`) from 1 - the number of matrices in the collection, or the name on page 78 (`String`) of the matrix.

Returns

`Matrix` on page 76

Remarks

Because `Item` is the default method for the `Matrices` collection, both of the following are acceptable:

```
.Matrices.Item(2)
.Matrices(2)
```

Examples

Retrieve the second matrix in the `Matrices` collection, name the matrix "gradient," then print the name in a message box.

```
Set MtbMatrix = MtbMatrices.Item(2)
MtbMatrix.Name = "gradient"
MsgBox "The second matrix is " & _
    MtbMatrix.Name
```

You can also retrieve the second matrix using this command:

```
Set MtbMatrix = Mtbmatrices(2)
```

Retrieve the matrix called "Gradient" and print the name in a message box.

```
Set MtbMatrix = MtbMatrices.Item("Gradient")
MsgBox "The current matrix is " & _
    MtbMatrix.Name
```

Matrices Collection method - Remove

Use to delete a `Matrix` object and remove it from the `Matrices` collection.

Syntax

```
Remove(Index as Variant)
```

Arguments

Index

Required. The index of the matrix as an integer (`Long`) from 1 - the number of matrices in the collection, or the name on page 78 (`String`) of the matrix.

Returns

HRESULT

Remarks

The same results can be achieved using the Delete on page 79 method of the Matrix on page 76 object. To remove all matrices, use the Delete on page 74 method of the Matrices on page 72 collection object.

Example

Remove the first matrix and the matrix named "Gradient1" from the Matrices collection.

```
MtbMatrices.Remove 1
MtbMatrices.Remove "Gradient1"
```

Matrix object

The Matrix object contains all the information related to an individual matrix. The Matrix object can contain *only* numeric data values.

Properties

Property	Description
ColumnCount on page 77	Number of columns in the Matrix object
Comment on page 77	Description of the Matrix object
MissingCount on page 78	Number of missing values in the Matrix object
Name on page 78	Name of the Matrix object
Number on page 78	Number of the Matrix object in the Matrices collection.
RowCount on page 79	Number of rows in the Matrix object

Methods

Method	Description
Delete on page 79	Use to delete a Matrix object and remove it from the Matrices collection.
GetData on page 80	Use to return the matrix value or values in the specified row and column.
SetData on page 80	Use to set values for a Matrix object using the elements specified in *Data*.

Example

Define and populate the array "arrTemps" with data values , retrieve the first matrix, name the matrix "Temperatures" and place the information in arrTemps into the "Temperatures" matrix with 4 rows and 3 columns, then adds a comment:

```
Dim arrTemps(1 To 12) As Single
    arrTemps(1) = 72
    arrTemps(2) = 95
    arrTemps(3) = 69
    arrTemps(4) = 87
```

```
    arrTemps(5) = 86
    arrTemps(6) = 75
    arrTemps(7) = 58
    arrTemps(8) = 92
    arrTemps(9) = 89
    arrTemps(10) = 66
    arrTemps(11) = 70
    arrTemps(12) = 91

Dim MtbMatrix As Mtb.Matrix
Set MtbMatrix = MtbMatrices(1)
MtbMatrix.Name = "Temperatures"
MtbMatrix.SetData arrTemps, 4, 3
MtbMatrix.Comment = "Temperatures for experiment 1"
```

Matrix property - ColumnCount

Description

Number of columns in the `Matrix` object

Type

Long

Range

1 - N

Access

Read-only

Example

Display in a message box the number of columns in the `Matrix` object.

```
MsgBox "This matrix has this many columns: " _
& MtbMatrix.ColumnCount
```

Matrix property - Comment

Description

Description of the `Matrix` object

Type

String

Range

Valid string

Access

Read/Write

Example

Add a comment to the matrix and print the comment in a message box.

```
MtbMatrix.Comment = "Temperature gradient values."
MsgBox MtbMatrix.Comment
```

Matrix property - MissingCount

Description
Number of missing values in the `Matrix` object

Type
Long

Range
N/A

Access
Read-only

Example

Display in a message box the number of missing values in the `Matrix` object.

```
MsgBox "This matrix has this many missing _
    values: " & MtbMatrix.MissingCount
```

Matrix property - Name

Description
Name of the `Matrix` object

Type
String

Range
Valid string

Access
Read/Write

Example

Retrieve the second matrix in the `Matrices` collection, name the matrix "Gradient 2," then print the name in a message box:

```
Set MtbMatrix = MtbMatrices(2)
MtbMatrix.Name = "Gradient 2"
MsgBox "The second matrix is " & _
    MtbMatrix.Name
```

Matrix property - Number

Description
Number of the `Matrix` object in the `Matrices` collection.

Type
Long

Range
1 - number of the Matrix objects in the Matrices collection (current Minitab limit is 1000)

Access
Read-only

Example

Display in a message box the number of the `Matrix` object within the `Matrices` collection.

```
MsgBox "This is matrix number " & _
    MtbMatrix.Number
```

Matrix property - RowCount

Description
Number of rows in the `Matrix` object

Type
Long

Range
1 - N

Access
Read-only

Example

Display in a message box the number of rows in the `Matrix` object.

```
MsgBox "This matrix has this many rows: " & _
    MtbMatrix.RowCount
```

Matrix method - Delete

Use to delete a `Matrix` object and remove it from the `Matrices` collection.

Syntax

```
Delete()
```

Returns

```
HRESULT
```

Remarks

The same results can be achieved using the `Remove` on page 75 method of the `Matrices` on page 72 Collection object. To delete all matrices, use the `Delete` on page 74 method of the `Matrices` on page 72 Collection object.

Example

Delete the `Matrix` object from the `Matrix` collection.

```
MtbMatrix.Delete
```

Matrix method - GetData

Use to return the matrix value or values in the specified row and column.

Syntax

```
GetData(Row as Long, Col as Long)
```

Arguments

Row
Optional. Row of the value to get.

Col
Optional. Column of the value to get.

Returns

```
Variant
```

Remarks

If you specify *Row* you must specify *Col*, and vice versa. If neither *Row* nor *Col* is specified, then `GetData` gets the entire matrix.

Multiple data values are returned in a vector in column major order, that is, all rows of column 1 are placed in the vector first, followed by column 2 rows, column 3 rows, etc.

Examples

Use the `GetData` method to populate a variant (safe array) with all the values from the current matrix. Then use a loop to print the values in the Immediate window.

```
Dim i As Integer
Dim cvTemp As Variant
cvTemp = MtbMatrix.GetData()

For i = 0 To 11
    Debug.Print cvTemp(i)
Next i
```

Get the value in the second row, second column of the matrix and print it in a message box.

```
MsgBox "The value is " & MtbMatrix.GetData(2,2)
```

Matrix method - SetData

Use to set values for a `Matrix` object using the elements specified in *Data*.

Syntax

```
SetData(Data as Variant, Rows as Long, Cols as Long)
```

Arguments

Data

Required. Numeric value or values to set in the matrix.

Rows

Required. Either the row of the matrix where a single value is to be set, or the number of rows to be set, starting at row 1.

Cols

Required. Either the column of the matrix where a single value is to be set, or the number of columns to be set, starting at column 1.

Returns

`HRESULT`

Remarks

If *Data* holds an individual numeric value, then *Rows* and *Cols* refer to an individual cell in the matrix that will be set. If *Data* holds multiple numeric values, then all previous data in the matrix is deleted and the matrix is set starting at position 1,1.

Data is read into the `Matrix` object column by column. Therefore, to set multiple data values in the matrix, *Data* must be a vector in column major order; that is, place all rows of column 1 in the vector first, followed by column 2 rows, column 3 rows, etc.

Examples

Retrieve the first matrix in the `Matrices` collection and populate three rows and eight columns in the matrix with the array "`arrIndex`."

```
Set MtbMatrix = MtbMatrices(1)
MtbMatrix.SetData(arrIndex, 3, 8)
```

Place the value 10 in row 20, column 4 of the current matrix, then print the value for row 20, column 4 in the Immediate window.

```
MtbMatrix.SetData(10, 20, 4)
Debug.Print MtbMatrix.GetData(20, 4)
```

C Command Object Reference

Commands Collection object

The `Commands` collection contains the commands that have been issued to Minitab during the session.

Properties

Property	Description
Count on page 83	Number of `Command` objects.
OutputDocument on page 83	Returns an `OutputDocument` object containing all output generated from all commands in the `Commands` collection.

Methods

Method	Description
Delete on page 83	Use to remove all `Command` objects from the `Commands` collection.
Item on page 84	Use to return a `Command` object within the `Commands` collection.
Remove on page 84	Use to delete a `Command` object and remove it from the `Commands` collection.

Example

Create a Minitab `Application` object and execute three Minitab commands. Then delete the first command, and loop through the remaining two, displaying a message box with the values of the following properties for each, as well as the name of the worksheet:

- `CommandLanguage`
- `Name`
- `Tag`
- `CreatedBy`
- `CreateDate`

Save the output document for each command as an RTF file and delete all commands at the end.

```
Dim MtbApp As New mtb.Application
Dim MtbProj As mtb.Project
Dim MtbCom As mtb.Command
Dim i, j As Integer

MtbApp.UserInterface.Visible = True
Set MtbProj = MtbApp.ActiveProject

MtbProj.ExecuteCommand "RANDOM 30 C1 - C2"
MtbProj.ExecuteCommand "REGRESS; RESPONSE C1; CONTINUOUS C2; TERMS C2."

'For the next command, use the ZTAG subcommand to set the Tag property.
MtbProj.ExecuteCommand "CORR C1 C2; ZTAG ""My Correlation""."
MtbProj.Commands.Item(1).Delete

For i = 1 To MtbProj.Commands.Count
```

```
    Set MtbCom = MtbProj.Commands(i)

    MsgBox "CommandLanguage = " & _
        MtbCom.CommandLanguage & vbCrLf & _
        "Command Name = " & MtbCom.Name & vbCrLf & _
        "Tag = " & MtbCom.Tag & vbCrLf & _
        "Created by " & MtbCom.CreatedBy & vbCrLf & _
        "Created on " & MtbCom.CreateDate & vbCrLf & _
        "Worksheet = " & MtbCom.Worksheet.Name

    MtbCom.OutputDocument.SaveAs _
        "C:\Output For Command " & i, True, OFRTF
Next i

MtbProj.Commands.Delete
```

Commands Collection property - Count

Description

Number of `Command` objects.

Type

Long

Range

Any valid long integer

Access

Read-only

Commands Collection property - OutputDocument

Description

Returns an `OutputDocument` object containing all output generated from all commands in the `Commands` collection.

Type

OutputDocument on page 99

Range

N/A

Access

Read-only

Commands Collection method - Delete

Use to remove all `Command` objects from the `Commands` collection.

Syntax

```
Delete()
```

Returns

HRESULT

Remarks

To remove a single command, use Remove on page 84 or the Delete on page 89 method of the Command on page 85 object.

Commands Collection method - Item

Use to return a Command object within the Commands collection.

Syntax

Item(*Index* as Variant)

Arguments

Index
Required. The index of the command as an integer (Long) from 1 - the number of commands in the collection.

Returns

Command on page 85

Remarks

Because Item is the default method for the Commands collection, both of the following are acceptable:

```
.Commands.Item(2)
.Commands(2)
```

Commands Collection method - Remove

Use to delete a Command object and remove it from the Commands collection.

Syntax

Remove(*Index* as Variant)

Arguments

Index
Required. The index of the command as an integer (Long) from 1 - the number of commands in the collection.

Returns

HRESULT

Remarks

The same results can be achieved using the `Delete` on page 89 method of the `Command` on page 85 object. To remove all commands, use the `Delete` on page 83 method of the `Commands` on page 82 collection object.

Example

Delete the first command from the `Commands` collection for a Minitab application set as `MtbApp`.

```
MtbApp.ActiveProject.Commands.Remove (1)
```

Command object

`Command` objects are created when you execute a Minitab command either programmatically or directly in Minitab.

Properties

Property	Description
CommandLanguage on page 86	Command language utilized to create the command object.
CreateDate on page 86	The date and time the command was generated.
CreatedBy on page 87	The user ID of the user who generated the command.
Name on page 87	The name of the Minitab command that generated the output.
OutputDocument on page 87	Returns an `OutputDocument` object containing all output generated by the command.
Outputs on page 88	Returns the `Outputs` collection generated by the command.
Tag on page 88	A string used to identify or describe the `Command` object.
Worksheet on page 88	The worksheet that was utilized as the input for the command.

Methods

Method	Description
Delete on page 89	Use to delete a `Command` object and remove it from the `Commands` collection.

Example

Create a Minitab `Application` object and execute three Minitab commands. Then delete the first command, and loop through the remaining two, displaying a message box with the values of the following properties for each, as well as the name of the worksheet:

- `CommandLanguage`
- `Name`
- `Tag`
- `CreatedBy`
- `CreateDate`

Save the output document for each command as an RTF file and delete all commands at the end.

```
Dim MtbApp As New mtb.Application
Dim MtbProj As mtb.Project
Dim MtbCom As mtb.Command
Dim i, j As Integer

MtbApp.UserInterface.Visible = True
Set MtbProj = MtbApp.ActiveProject

MtbProj.ExecuteCommand "RANDOM 30 C1 - C2"
MtbProj.ExecuteCommand "REGRESS; RESPONSE C1; CONTINUOUS C2; TERMS C2."

'For the next command, use the ZTAG subcommand to set the Tag property.
MtbProj.ExecuteCommand "CORR C1 C2; ZTAG ""My Correlation""."
MtbProj.Commands.Item(1).Delete

For i = 1 To MtbProj.Commands.Count
    Set MtbCom = MtbProj.Commands(i)

    MsgBox "CommandLanguage = " & _
        MtbCom.CommandLanguage & vbCrLf & _
        "Command Name = " & MtbCom.Name & vbCrLf & _
        "Tag = " & MtbCom.Tag & vbCrLf & _
        "Created by " & MtbCom.CreatedBy & vbCrLf & _
        "Created on " & MtbCom.CreateDate & vbCrLf & _
        "Worksheet = " & MtbCom.Worksheet.Name

    MtbCom.OutputDocument.SaveAs _
        "C:\Output For Command " & i, True, OFRTF
Next i

MtbProj.Commands.Delete
```

Command property - CommandLanguage

Description

Command language utilized to create the command object.

Type

String

Range

Valid string

Access

Read-only

If the command is a custom command, the value of the `CommandLanguage` property is "COMCUSTOM."

Command property - CreateDate

Description

The date and time the command was generated.

Type

String

Range

Valid string

Access
Read-only

Command property - CreatedBy

Description
The user ID of the user who generated the command.

Type
String

Range
Valid string

Access
Read-only

Command property - Name

Description
The name of the Minitab command that generated the output.

Type
String

Range
Valid string

Access
Read-only

Command property - OutputDocument

Description
Returns an `OutputDocument` object containing all output generated by the command.

Type
OutputDocument on page 99

Range
N/A

Access
Read-only

Command property - Outputs

Description
Returns the `Outputs` collection generated by the command.

Type
Outputs on page 89 collection

Range
N/A

Access
Read-only

Command property - Tag

Description
A string used to identify or describe the `Command` object. Null by default.

Type
String

Range
Valid string

Access
Read/Write

The `Tag` property for most commands can also be set from the Minitab itself using the `ZTAG` subcommand. For example, entering the following in Minitab's Session window or Command Line Editor creates a `Command` object with the tag "My Z-Test." The argument for `ZTAG` may be a string or text constant.

```
MTB > OneZ 20 3;
SUBC>   Sigma 1;
SUBC>   Test 2;
SUBC>   ZTAG "My Z-Test".
```

Command property - Worksheet

Description
The worksheet that was utilized as the input for the command.

Type
Worksheet on page 43

Range
Any worksheet in the `Worksheets` collection

Access
Read-only for Minitab commands; Read/Write for custom commands

Command method - Delete

Use to delete a `Command` object and remove it from the `Commands` collection.

Syntax

```
Delete()
```

Returns

```
HRESULT
```

Remarks

The same results can be achieved using the Remove on page 84 method of the `Commands` on page 82 collection.

Outputs Collection object

The `Outputs` collection for each `Command` on page 85 object contains all the output generated by that command.

Properties

Property	Description
Count on page 90	The number of `Output` on page 92 objects within the `Outputs` collection.

Methods

Method	Description
Delete on page 90	Use to remove all `Output` objects from the `Outputs` collection.
Item on page 91	Use to return an `Output` object within the `Outputs` collection.
OpenGraph on page 91	Use to open a Minitab graph file and add it as a `Graph` object to the `Outputs` collection.
Remove on page 91	Use to delete an `Output` object and remove it from the `Outputs` collection.

Example

Generate random data, then create a scatterplot and a regression analysis. Save the scatterplot to a file, then add it to the outputs collection of the regression command. Delete `Output` objects using the `Remove` method of the `Outputs` collection and the `Delete` method of an `Output` object. Finally, delete all outputs for a command at once using the `Delete` method of the `Outputs` collection.

```
Dim MtbApp As New Mtb.Application
Dim MtbProj As Mtb.Project

MtbApp.UserInterface.Visible = True
Set MtbProj = MtbApp.ActiveProject

With MtbProj
    .ExecuteCommand "RANDOM 30 C1 - C2"
    .ExecuteCommand "PLOT C1*C2"
```

```
    .ExecuteCommand "REGRESS; RESPONSE C1; CONTINUOUS C2; TERMS C2."
End With

'Save the plot to a file
MtbProj.Commands(2).Outputs.Item(1).Graph.SaveAs "C:\MyGraph", True, GFMinitab
With MtbProj.Commands(3)

'Add the saved graph to the collection
    .Outputs.OpenGraph "C:\MyGraph.MGF"

'Save the output document as an HTML file
    .OutputDocument.SaveAs "C:\MyOutput", True, OFHTML

'Delete the first 2 Output objects and save the output document again
    .Outputs.Remove 1
    .Outputs(1).Delete
    .OutputDocument.SaveAs "C:\MyOutput2", True, OFHTML

'Delete all remaining Outputs
    .Outputs.Delete
End With
```

Outputs Collection property - Count

Description

The number of `Output` on page 92 objects within the `Outputs` collection.

Type

Long

Range

0 to the number of `Output` objects within the `Outputs` collection

Access

Read-only

Outputs Collection method - Delete

Use to remove all `Output` objects from the `Outputs` collection.

Syntax

```
Delete()
```

Returns

```
HRESULT
```

Remarks

To remove a single Output object, use `Remove` on page 91 or the `Delete` on page 98 method of the `Output` on page 92 object.

Outputs Collection method - Item

Use to return an `Output` object within the `Outputs` collection.

Syntax

```
Item(Index as Variant)
```

Arguments

Index
Required. The index of the Output object as an integer (`Long`) from 1 - the number of `Output` objects in the collection.

Returns

`Output` on page 92

Remarks

Because `Item` is the default method for the `Outputs` collection, both of the following are acceptable:

```
.Outputs.Item(2)
.Outputs(2)
```

Outputs Collection method - OpenGraph

Use to open a Minitab graph file and add it as a `Graph` object to the `Outputs` collection.

Syntax

```
OpenGraph(Filename as String)
```

Arguments

Filename
Optional. The path and name of the graph file to be opened. If a path is not specified, the `DefaultFilePath` on page 24 is used. If an extension is not specified the default extension .MGF is used.

Remarks

The new graph is returned.

Outputs Collection method - Remove

Use to delete an `Output` object and remove it from the `Outputs` collection.

Syntax

```
Remove(Index as Variant)
```

Arguments

Index

Required. The index of the output as an integer (`Long`) from 1 - the number of outputs in the collection.

Returns

`HRESULT`

Remarks

The same results can be achieved using the `Delete` on page 98 method of the `Output` on page 92 object. To remove all `Output` objects, use the `Delete` on page 65 method of the `Outputs` on page 89 collection.

Output object

Each `Output` object contains one component of the output from a Minitab Command on page 85 object.

The `OutputType` and `Tag` properties are universal and apply to all `Output` objects. Each of the remaining properties are valid only for one specific output type. For example, using the `Formula` property on an `Output` object of type `OTFormula` on page 8, returns a `Formula` object. Using the `Formula` property on any other `MtbOutputTypes` on page 8 returns an error.

Properties

Property	Description
Formula on page 94	If the `Output` object is a formula, this property returns the corresponding `Formula` object.
Graph on page 95	If the `Output` object is a graph, this property returns the corresponding `Graph` object.
HTMLText on page 95	The contents of the `Output` object as HTML-formatted text.
Message on page 95	If the `Output` object is a message, this property returns the corresponding `Message` object.
OutList on page 96	If the `Output` object is an `OutList`, this property returns the corresponding `OutList` object.
OutputType on page 96	The `Output` object type.
Paragraph on page 96	If the `Output` object is a paragraph, this property returns the corresponding `Paragraph` object.
RTFText on page 96	The contents of the `Output` object as RTF-formatted text.
Table on page 97	If the `Output` object is a table, this property returns the corresponding `Table` object.
Tag on page 97	A string used to identify or describe the `Output` object.
Text on page 97	The contents of the `Output` object as un-formatted text.
Title on page 98	If the `Output` object is a title, this property returns the corresponding `Title` object.

Property	Description
Triangle on page 98	If the `Output` object is a triangle, this property returns the corresponding `Triangle` object.

Methods

Method	Description
Delete on page 98	Use to delete an `Output` object and remove it from the `Outputs` collection.

Examples

Create a Minitab `Application` object and execute several Minitab commands. Loop through all `Output` objects in the `Outputs` collection for each `Command`, using the `OutputType` property to identify each output type. Display a message for each `Output` object stating the `Index` number of the `Command`, as well as the `Index` number, `OutputType`, and the `Text` of the `Output` object. Finally, save all output for all commands as an HTML file, using the `OutputDocument` object of the `Commands` collection.

```
Dim MtbApp As New mtb.Application
Dim MtbProj As mtb.Project
Dim MtbCmnd As mtb.Command
Dim MtbOutObj As mtb.Output
Dim i, j As Integer
Dim MsgStr As String

MtbApp.UserInterface.Visible = True
Set MtbProj = MtbApp.ActiveProject

'Execute some Minitab commands
With MtbProj
  .ExecuteCommand "RANDOM 30 C1-C3"
  .ExecuteCommand "PLOT C1*C2"
  .ExecuteCommand "REGRESS; RESPONSE C1; CONTINUOUS C2; TERMS C2."
  .ExecuteCommand "CORR C1 C2 C3 C4"
  'This CORR command will generate a correlation analysis_
    and also an error message because there is no data in C4

  'Add a worksheet and create a DOE design
  .Worksheets.Add
  .ExecuteCommand "FFDESIGN 4 8; CTPT c3; RANDOMIZE; " & _
    "SORDER C1 C2; BRIEF 4; ALIAS 4; XMATRIX C5 C6 C7 C8."
End With

'Loop through outputs from commands, identify type of each, _
  and display message
For i = 1 To MtbProj.Commands.Count
  Set MtbCmnd = MtbProj.Commands(i)
  For j = 1 To MtbCmnd.Outputs.Count
    Set MtbOutObj = MtbCmnd.Outputs(j)
    MsgStr = "Command #" & i & ", " & "Output #" & j & _
      " is OutputType "
    Select Case MtbOutObj.OutputType
      Case 0 'Graph
        MsgBox MsgStr & _
          "Graph."
      Case 1 'Table
        MsgBox MsgStr & _
          "Table with the following text:" & _
          vbCrLf & _
          MtbOutObj.Table.Text
      Case 2 'OutList
        MsgBox MsgStr & _
          "OutList with the following text:" & _
          vbCrLf & _
```

```
            MtbOutObj.OutList.Text
        Case 3 'Title
          MsgBox MsgStr & _
            "Title with the following text:" & _
            vbCrLf & _
            MtbOutObj.Title.Text
        Case 4 'Message
          MsgBox MsgStr & _
            "Message with the following text:" & _
            vbCrLf & _
            MtbOutObj.Message.Text
        Case 5 'Paragraph
          MsgBox MsgStr & _
            "Paragraph with the following text:" & _
            vbCrLf & _
            MtbOutObj.Paragraph.Text
        Case 6 'Formula
          MsgBox MsgStr & _
            "Formula with the following text:" & _
            vbCrLf & _
            MtbOutObj.Formula.Text
        Case 7 'Triangle
          MsgBox MsgStr & _
            "Triangle with the following text:" & _
            vbCrLf & _
            MtbOutObj.Triangle.Text
      End Select
    Next
Next

'Save OutputDocument as an HTM file named mtb_out.htm
MtbProj.Commands.OutputDocument.SaveAs _
  "c:\mtb_out", True
```

Create a new instance of Minitab, generate two columns of random data, and run a correlation analysis. Display the `Text`, `RTFText`, and `HTMLText` for each output of the analysis in a message box.

```
Dim MtbApp As New Mtb.Application
Dim MtbOuts As Mtb.Outputs
Dim MtbOut As Mtb.Output
Dim i As Integer
MtbApp.UserInterface.Visible = True
With MtbApp
    With .ActiveProject
        .ExecuteCommand "RANDOM 30 C1-C2"
        .ExecuteCommand "CORRELATION C1 C2"
        Set MtbOuts = .Commands(2).Outputs
    End With

    For i = 1 To MtbOuts.Count
        Set MtbOut = MtbOuts.Item(i)
        MsgBox ("Text for Output " & i & ":" & vbCrLf & vbCrLf & MtbOut.Text)
        MsgBox ("RTFText for Output " & i & ":" & vbCrLf & vbCrLf & MtbOut.RTFText)
        MsgBox ("HTMLText for Output " & i & ":" & vbCrLf & vbCrLf & MtbOut.HTMLText)
    Next i
End With
```

Output property - Formula

Description

If the `Output` object is a formula, this property returns the corresponding `Formula` object. Otherwise, an error is generated.

Type

Formula on page 106

Range
N/A

Access
Read-only

Output property - Graph

Description
If the `Output` object is a graph, this property returns the corresponding `Graph` object. Otherwise, an error is generated.

Type
Graph on page 112

Range
N/A

Access
Read-only

Output property - HTMLText

Description
The contents of the `Output` object as HTML-formatted text.

Type
String

Range
Valid string

Access
Read-only

Output property - Message

Description
If the `Output` object is a message, this property returns the corresponding `Message` object. Otherwise, an error is generated.

Type
Message on page 117

Range
N/A

Access
Read-only

Output property - OutList

Description
If the `Output` object is an `OutList`, this property returns the corresponding `OutList` object. Otherwise, an error is generated.

Type
OutList on page 114

Range
N/A

Access
Read-only

Output property - OutputType

Description
The `Output` object type.

Type
MtbOutputTypes on page 8

Range
Any `MtbOutputTypes` constant

Access
Read-only

Output property - Paragraph

Description
If the `Output` object is a paragraph, this property returns the corresponding `Paragraph` object. Otherwise, an error is generated.

Type
Paragraph on page 103

Range
N/A

Access
Read-only

Output property - RTFText

Description
The contents of the `Output` object as RTF-formatted text.

Type
String

Range
Valid string

Access
Read-only

Output property - Table

Description
If the `Output` object is a table, this property returns the corresponding `Table` object. Otherwise, an error is generated.

Type
Table on page 105

Range
N/A

Access
Read-only

Output property - Tag

Description
A string used to identify or describe the `Output` object. Null by default.

Type
String

Range
Valid string

Access
Read/Write

Example

Set the `Tag` text for the first `Output` object in `MtbCommand` to "This is Output number 1," then display the tag in a message box.

```
MtbCommand.Outputs(1).Tag = "This is Output number 1"
MsgBox MtbCommand.Outputs(1).Tag
```

Output property - Text

Description
The contents of the `Output` object as un-formatted text.

Type
String

Range
Valid string

Access
Read-only

Output property - Title

Description
If the `Output` object is a title, this property returns the corresponding `Title` object. Otherwise, an error is generated.

Type
Title on page 102

Range
N/A

Access
Read-only

Output property - Triangle

Description
If the `Output` object is a triangle, this property returns the corresponding `Triangle` object. Otherwise, an error is generated.

Type
Triangle on page 116

Range
N/A

Access
Read-only

Output method - Delete

Use to delete an `Output` object and remove it from the `Outputs` collection.

Syntax

```
Delete()
```

Returns

```
HRESULT
```

Example

Delete the first `Output` object from the command set as `MtbCommand`.

```
MtbCommand.Outputs(1).Delete
```

Remarks

The same results can be achieved using the `Remove` on page 91 method of the `Outputs` on page 89 collection.

OutputDocument object

An `OutputDocument` object contains all output generated by a single `Command` on page 85 object or by all commands in the `Commands` on page 82 collection.

Properties

Property	Description
HTMLText on page 100	Content of `OutputDocument` in HTML format.
OutputWidth on page 100	The output width in characters.
RTFText on page 100	Content of `OutputDocument` in RTF format.
Text on page 101	Content of `OutputDocument` as plain text.

Methods

Method	Description
CopyToClipboard on page 101	Use to copy the `OutputDocument` object to the Windows clipboard.
PrintOut on page 101	Use to send an `OutputDocument` object to the ActivePrinter on page 27.
SaveAs on page 101	Use to save a copy of the `OutputDocument` object.

Example

This example creates a Minitab `Application` object and executes some Minitab commands. It then prints out the `OutputDocument` for the `Commands` collection, saves it, and copies it to the Windows clipboard. Then the content of the `OutputDocument` is saved in string variables as text, HTML formatted text, and RTF formatted text.

```
Dim MtbApp As New Mtb.Application
Dim MtbProj As Mtb.Project
Dim MtbOutDoc As Mtb.OutputDocument
Dim sText, sHTML, sRTF As String

MtbApp.UserInterface.Visible = True
Set MtbProj = MtbApp.ActiveProject

With MtbProj
    .ExecuteCommand "RANDOM 30 C1"
    .ExecuteCommand "DESCRIBE C1"
End With

Set MtbOutDoc = MtbProj.Commands.OutputDocument

With MtbOutDoc
```

```
    .OutputWidth = 50
    .PrintOut
    .SaveAs "C:\MyOutputDocument", True, OFRTF
    .CopyToClipboard
    sText = .Text
    sHTML = .HTMLText
    sRTF = .RTFText
End With
```

OutputDocument property - HTMLText

Description

Content of `OutputDocument` in HTML format.

Type

String

Range

Valid string

Access

Read-only

OutputDocument property - OutputWidth

Description

The output width in characters. The default is 80.

Type

Double

Range

> 0

Access

Read/Write

OutputDocument property - RTFText

Description

Content of `OutputDocument` in RTF format.

Type

String

Range

Valid string

Access

Read-only

OutputDocument property - Text

Description
Content of `OutputDocument` as plain text.

Type
String

Range
Valid string

Access
Read-only

OutputDocument method - CopyToClipboard

Use to copy the `OutputDocument` object to the Windows clipboard.

Syntax

```
CopyToClipboard()
```

Returns

```
HRESULT
```

OutputDocument method - PrintOut

Use to send an `OutputDocument` object to the ActivePrinter on page 27.

Syntax

```
PrintOut()
```

Returns

```
HRESULT
```

OutputDocument method - SaveAs

Use to save a copy of the `OutputDocument` object.

Syntax

```
SaveAs(Filename as String, Replace as Boolean, OutputFileType as MtbOutputFileTypes)
```

Arguments

Filename
Required. Path and file name to use when saving the file. If a path is not specified, then the DefaultFilePath on page 24 is used.

Replace
Optional. If True, an existing file with the same name will be overwritten. The default is False.

OutputFileType
Optional. The format to use when saving the file. May be any MtbOutputFileTypes on page 8 constant.

Returns

HRESULT

Remarks

If you don't specify an extension matching the file type, the appropriate one (.HTM or .RTF) is automatically added to the file name.

Title object

Each Title object contains a single title generated by a Minitab Command on page 85 object.

Properties

Property	Description
HTMLText on page 103	The contents of the Title object as HTML-formatted text.
RTFText on page 103	The contents of the Title object as RTF-formatted text.
Text on page 103	The contents of the Title object as unformatted text.

Example

Create a new Minitab Application object, execute a command that generates a Title object as Output object number 1, retrieve the Title, and display messages showing its Text, HTMLText, and RTFText properties.

```
Dim MtbApp As New Mtb.Application
Dim MtbProj As Mtb.Project
Dim MtbTitle As Mtb.Title

MtbApp.UserInterface.Visible = True
Set MtbProj = MtbApp.ActiveProject
MtbProj.ExecuteCommand "RANDOM 30 C1 - C3"
MtbProj.ExecuteCommand "REGRESS; RESPONSE C1; CONTINUOUS C2; TERMS C2."

Set MtbTitle = MtbProj.Commands(2).Outputs(1).Title

MsgBox "Title text: " & vbCrLf & _
    MtbTitle.Text
MsgBox "Title HTMLText: " & vbCrLf & _
    MtbTitle.HTMLText
```

```
MsgBox "Title RTFText: " & vbCrLf & _
    MtbTitle.RTFText
```

Title property - HTMLText

Description
The contents of the `Title` object as HTML-formatted text.

Type
String

Range
Valid string

Access
Read-only

Title property - RTFText

Description
The contents of the `Title` object as RTF-formatted text.

Type
String

Range
Valid string

Access
Read-only

Title property - Text

Description
The contents of the `Title` object as unformatted text.

Type
String

Range
Valid string

Access
Read-only

Paragraph object

Each `Paragraph` object contains a single output paragraph generated by a Minitab `Command` on page 85 object.

Properties

Property	Description
HTMLText on page 104	The contents of the `Paragraph` object as HTML-formatted text.
RTFText on page 104	The contents of the `Paragraph` object as RTF-formatted text.
Text on page 105	The contents of the `Paragraph` object as unformatted text.

Example

Create a new Minitab `Application` object, execute a command that generates a `Paragraph` object as `Output` object number 4, retrieve the `Paragraph` object, and display messages showing its `Text`, `HTMLText`, and `RTFText` properties.

```
Dim MtbApp As New Mtb.Application
Dim MtbProj As Mtb.Project
Dim MtbPar As Mtb.Paragraph

MtbApp.UserInterface.Visible = True
Set MtbProj = MtbApp.ActiveProject
MtbProj.ExecuteCommand "RANDOM 30 C1 - C3"
MtbProj.ExecuteCommand "REGRESS; RESPONSE C1; CONTINUOUS C2; TERMS C2."

Set MtbPar = MtbProj.Commands(2).Outputs(4).Paragraph

MsgBox "Paragraph text: " & vbCrLf & _
    MtbPar.Text
MsgBox "Paragraph HTMLText: " & vbCrLf & _
    MtbPar.HTMLText
MsgBox "Paragraph RTFText: " & vbCrLf & _
    MtbPar.RTFText
```

Paragraph property - HTMLText

Description

The contents of the `Paragraph` object as HTML-formatted text.

Type

String

Range

Valid string

Access

Read-only

Paragraph property - RTFText

Description

The contents of the `Paragraph` object as RTF-formatted text.

Type

String

Range
Valid string

Access
Read-only

Paragraph property - Text

Description
The contents of the `Paragraph` object as unformatted text.

Type
String

Range
Valid string

Access
Read-only

Table object

Each `Table` object contains a single output table generated by a Minitab `Command` on page 85 object.

Properties

Property	Description
HTMLText on page 106	The contents of the `Table` object as HTML-formatted text.
RTFText on page 106	The contents of the `Table` object as RTF-formatted text.
Text on page 106	The contents of the `Table` object as unformatted text.

Example

Create a new Minitab `Application` object, execute a command that generates a `Table` object as `Output` object number 3, retrieve the `Table` object, and display messages showing its `Text`, `HTMLText`, and `RTFText` properties.

```
Dim MtbApp As New Mtb.Application
Dim MtbProj As Mtb.Project
Dim MtbTable As Mtb.Table

MtbApp.UserInterface.Visible = True
Set MtbProj = MtbApp.ActiveProject
MtbProj.ExecuteCommand "RANDOM 30 C1 - C2"
MtbProj.ExecuteCommand "REGRESS; RESPONSE C1; CONTINUOUS C2; TERMS C2."

Set MtbTable = MtbProj.Commands(2).Outputs(3).Table

MsgBox "Text of Table:" & vbCrLf & _
    MtbTable.Text

MsgBox "HTMLText of Table:" & vbCrLf & _
    MtbTable.HTMLText
```

```
MsgBox "RTFText of Table:" & vbCrLf & _
    MtbTable.RTFText
```

Table property - HTMLText

Description
The contents of the `Table` object as HTML-formatted text.

Type
String

Range
Valid string

Access
Read-only

Table property - RTFText

Description
The contents of the `Table` object as RTF-formatted text.

Type
String

Range
Valid string

Access
Read-only

Table property - Text

Description
The contents of the `Table` object as unformatted text.

Type
String

Range
Valid string

Access
Read-only

Formula object

Each `Formula` object contains a single output formula generated by a Minitab `Command` on page 85 object.

Properties

Property	Description
Coefficients on page 107	Returns a `FormulaCoefficients` collection.
HTMLText on page 108	The contents of the `Formula` object as HTML-formatted text.
Response on page 108	The label for the response factor in the formula.
RTFText on page 108	The contents of the `Formula` object as RTF-formatted text.
Text on page 109	The contents of the `Formula` object as unformatted text.

Example

Create a new Minitab `Application` object, execute a command that generates a `Formula` object as `Output` object number 2, retrieve the `Formula` object, and display messages showing its `Text`, `HTMLText`, and `RTFText` properties. Then loop through the `FormulaCoefficients` collection for the formula and display a message for each `FormulaCoefficient` object showing its `Label`, `Value`, and `StandardError` properties.

```
Dim MtbApp As New Mtb.Application
Dim MtbProj As Mtb.Project
Dim MtbForm As Mtb.Formula
Dim MtbCoef As Mtb.FormulaCoefficient
Dim i As Integer

MtbApp.UserInterface.Visible = True
Set MtbProj = MtbApp.ActiveProject
MtbProj.ExecuteCommand "RANDOM 30 C1 - C3"
MtbProj.ExecuteCommand "REGRESS; RESPONSE C1; CONTINUOUS C2 C3; TERMS C2 C3."

Set MtbForm = MtbProj.Commands(2).Outputs(2).Formula

MsgBox "Formula text: " & vbCrLf & _
    MtbForm.Text
MsgBox "Formula HTMLText: " & vbCrLf & _
    MtbForm.HTMLText
MsgBox "Formula RTFText: " & vbCrLf & _
    MtbForm.RTFText
MsgBox "Response column: " & vbCrLf & _
    MtbForm.Response

For i = 1 To MtbForm.Coefficients.Count
    Set MtbCoef = MtbForm.Coefficients(i)
    MsgBox "Coefficient #" & i & vbCrLf & _
        "Label: " & MtbCoef.Label & vbCrLf & _
        "Value: " & MtbCoef.Value & vbCrLf & _
        "Standard Error: " & MtbCoef.StandardError
Next i
```

Formula property - Coefficients

Description

Returns a `FormulaCoefficients` collection.

Type

FormulaCoefficients on page 109

Range

N/A

Access
Read-only

Formula property - HTMLText

Description
The contents of the `Formula` object as HTML-formatted text.

Type
String

Range
Valid string

Access
Read-only

Formula property - Response

Description
The label for the response factor in the formula.

Type
String

Range
Valid string

Access
Read-only

Formula property - RTFText

Description
The contents of the `Formula` object as RTF-formatted text.

Type
String

Range
Valid string

Access
Read-only

Formula property - Text

Description
The contents of the `Formula` object as unformatted text.

Type
String

Range
Valid string

Access
Read-only

FormulaCoefficients Collection object

The `FormulaCoefficients` collection contains all the coefficients from a formula.

Properties

Property	Description
Count on page 110	The number of `FormulaCoefficient` objects within the `FormulaCoefficients` collection.

Methods

Method	Description
Item on page 110	Use to return a `FormulaCoefficient` object within the `FormulaCoefficients` collection.

Example

Create a new Minitab `Application` object, execute a command that generates a `Formula` object as `Output` object number 2, retrieve the `Formula` object, and display messages showing its `Text`, `HTMLText`, and `RTFText` properties. Then loop through the `FormulaCoefficients` collection for the formula and display a message for each `FormulaCoefficient` object showing its `Label`, `Value`, and `StandardError` properties.

```
Dim MtbApp As New Mtb.Application
Dim MtbProj As Mtb.Project
Dim MtbForm As Mtb.Formula
Dim MtbCoef As Mtb.FormulaCoefficient
Dim i As Integer

MtbApp.UserInterface.Visible = True
Set MtbProj = MtbApp.ActiveProject
MtbProj.ExecuteCommand "RANDOM 30 C1 - C3"
MtbProj.ExecuteCommand "REGRESS; RESPONSE C1; CONTINUOUS C2 C3; TERMS C2 C3."

Set MtbForm = MtbProj.Commands(2).Outputs(2).Formula

MsgBox "Formula text: " & vbCrLf & _
    MtbForm.Text
MsgBox "Formula HTMLText: " & vbCrLf & _
```

```
    MtbForm.HTMLText
MsgBox "Formula RTFText: " & vbCrLf & _
    MtbForm.RTFText
MsgBox "Response column: " & vbCrLf & _
    MtbForm.Response

For i = 1 To MtbForm.Coefficients.Count
    Set MtbCoef = MtbForm.Coefficients(i)
    MsgBox "Coefficient #" & i & vbCrLf & _
        "Label: " & MtbCoef.Label & vbCrLf & _
        "Value: " & MtbCoef.Value & vbCrLf & _
        "Standard Error: " & MtbCoef.StandardError
Next i
```

FormulaCoefficients Collection property - Count

Description

The number of `FormulaCoefficient` objects within the `FormulaCoefficients` collection.

Type

Long

Range

0 - the number of `FormulaCoefficient` on page 111 objects within the `FormulaCoefficients` collection

Access

Read-only

FormulaCoefficients Collection method - Item

Use to return a `FormulaCoefficient` object within the `FormulaCoefficients` collection.

Syntax

`Item(`*`Index`* `as Long)`

Arguments

Index

Required. The index of the `FormulaCoefficient` as an integer (`Long`) from 1 - the number of `FormulaCoefficients` in the collection.

Returns

`FormulaCoefficient` on page 111

Remarks

Because Item is the default method for the `FormulaCoefficients` collection, both of the following are acceptable:

```
.FormulaCoefficients.Item(2)
.FormulaCoefficients(2)
```

FormulaCoefficient object

Each `FormulaCoefficient` object contains a single coefficient.

Properties

Property	Description
Label on page 111	The label given the coefficient.
StandardError on page 112	The standard error of the coefficient.
Value on page 112	The numeric value of the coefficient.

Example

Create a new Minitab `Application` object, execute a command that generates a `Formula` object as `Output` object number 2, retrieve the `Formula` object, and display messages showing its `Text`, `HTMLText`, and `RTFText` properties. Then loop through the `FormulaCoefficients` collection for the formula and display a message for each `FormulaCoefficient` object showing its `Label`, `Value`, and `StandardError` properties.

```
Dim MtbApp As New Mtb.Application
Dim MtbProj As Mtb.Project
Dim MtbForm As Mtb.Formula
Dim MtbCoef As Mtb.FormulaCoefficient
Dim i As Integer

MtbApp.UserInterface.Visible = True
Set MtbProj = MtbApp.ActiveProject
MtbProj.ExecuteCommand "RANDOM 30 C1 - C3"
MtbProj.ExecuteCommand "REGRESS; RESPONSE C1; CONTINUOUS C2 C3; TERMS C2 C3."

Set MtbForm = MtbProj.Commands(2).Outputs(2).Formula

MsgBox "Formula text: " & vbCrLf & _
    MtbForm.Text
MsgBox "Formula HTMLText: " & vbCrLf & _
    MtbForm.HTMLText
MsgBox "Formula RTFText: " & vbCrLf & _
    MtbForm.RTFText
MsgBox "Response column: " & vbCrLf & _
    MtbForm.Response

For i = 1 To MtbForm.Coefficients.Count
    Set MtbCoef = MtbForm.Coefficients(i)
    MsgBox "Coefficient #" & i & vbCrLf & _
        "Label: " & MtbCoef.Label & vbCrLf & _
        "Value: " & MtbCoef.Value & vbCrLf & _
        "Standard Error: " & MtbCoef.StandardError
Next i
```

FormulaCoefficient property - Label

Description

The label given the coefficient.

Type

String

Range
Valid string

Access
Read-only

FormulaCoefficient property - StandardError

Description
The standard error of the coefficient.

Type
Double

Range
Any valid double-precision value

Access
Read-only

FormulaCoefficient property - Value

Description
The numeric value of the coefficient.

Type
Double

Range
Any valid double-precision value

Access
Read-only

Graph object

Each `Graph` object contains a single graph generated by a Minitab `Command` on page 85 object.

Methods

Method	Description
CopyToClipboard on page 113	Use to copy the `Graph` object to the Windows clipboard.
PrintOut on page 113	Use to send a `Graph` object to the ActivePrinter on page 27.
SaveAs on page 113	Use to save a copy of the `Graph` object.

Example

Create a new Minitab `Application` object, execute a command that generates a `Graph` object as `Output` object number 1, retrieve the graph, print it to the active printer on page 27, and then save it as a color PNG image file. Finally, copy the graph to the system clipboard.

```
Dim MtbApp As New Mtb.Application
Dim MtbProj As Mtb.Project
Dim MtbGraph As Mtb.Graph

MtbApp.UserInterface.Visible = True
Set MtbProj = MtbApp.ActiveProject
MtbProj.ExecuteCommand "RANDOM 30 C1"
MtbProj.ExecuteCommand "HISTOGRM C1"

Set MtbGraph = MtbProj.Commands(2).Outputs(1).Graph

MtbGraph.PrintOut
MtbGraph.SaveAs "C:\MyGraph", True, GFPNGColor
MtbGraph.CopyToClipboard
```

Graph method - CopyToClipboard

Use to copy the `Graph` object to the Windows clipboard.

Syntax

```
CopyToClipboard()
```

Returns

```
HRESULT
```

Remarks

You can copy a Minitab `Graph` object, Bitmap, MetaFile, or Enhanced metafile. For Minitab `Graph` objects, either a paste or a paste link operation is allowed with the copied object. For all other file formats, only paste is allowed.

Graph method - PrintOut

Use to send a `Graph` object to the ActivePrinter on page 27.

Syntax

```
PrintOut()
```

Returns

```
HRESULT
```

Graph method - SaveAs

Use to save a copy of the `Graph` object.

Syntax

```
SaveAs(Filename as String, Replace as Boolean, GraphFileType as MtbGraphFileTypes,
Width as Long, Height as Long)
```

Arguments

Filename
Optional. Path and file name to use when saving the graph. If a path is not specified, then the DefaultFilePath on page 24 is used. The default file name is Minitab.

Replace
Optional. If True, an existing file with the same name will be overwritten. The default is True.

GraphFileType
Optional. The format to use when saving the file. May be any MtbGraphFileTypes on page 7 constant. The default is .MGF.

Width
Optional. Use to set the width of the graph in pixels.

Height
Optional. Use to set the height of the graph in pixels.

Returns

HRESULT

Remarks

If you don't specify an extension matching the file type, the appropriate one (.MGF, .JPG, .PNG, .TIF, or .BMP) is automatically added to the file name.

OutList object

Each OutList object contains a single OutList generated by a Minitab Command on page 85 object.

Properties

Property	Description
HTMLText on page 115	The contents of the OutList object as HTML-formatted text.
RTFText on page 115	The contents of the OutList object as RTF-formatted text.
Text on page 116	The contents of the OutList object as unformatted text.

Example

Create a new Minitab `Application` object, execute a command that generates an `OutList` object as `Output` object number 3, retrieve the `OutList` object, and display messages showing its `Text`, `HTMLText`, and `RTFText` properties.

```
Dim MtbApp As New Mtb.Application
Dim MtbProj As Mtb.Project
Dim MtbOutList As Mtb.OutList

MtbApp.UserInterface.Visible = True
Set MtbProj = MtbApp.ActiveProject
MtbProj.ExecuteCommand _
    "FFDESIGN 4 8; CTPT c3; RANDOMIZE; " & _
    "SORDER C1 C2; BRIEF 4; ALIAS 4; XMATRIX C5 C6 C7 C8."

Set MtbOutList = MtbProj.Commands(1).Outputs(3).OutList

MsgBox "OutList text: " & vbCrLf & _
    MtbOutList.Text
MsgBox "OutList HTMLText: " & vbCrLf & _
    MtbOutList.HTMLText
MsgBox "OutList RTFText: " & vbCrLf & _
    MtbOutList.RTFText
```

OutList property - HTMLText

Description

The contents of the `OutList` object as HTML-formatted text.

Type

String

Range

Valid string

Access

Read-only

OutList property - RTFText

Description

The contents of the `OutList` object as RTF-formatted text.

Type

String

Range

Valid string

Access

Read-only

OutList property - Text

Description
The contents of the `OutList` object as unformatted text.

Type
String

Range
Valid string

Access
Read-only

Triangle object

Each `Triangle` object contains a single output triangle generated by a Minitab `Command` on page 85 object.

Properties

Property	Description
HTMLText on page 116	The contents of the `Triangle` object as HTML-formatted text.
RTFText on page 117	The contents of the `Triangle` object as RTF-formatted text.
Text on page 117	The contents of the `Triangle` object as unformatted text.

Example

Create a new Minitab `Application` object, execute a command that generates a `Triangle` object as `Output` object number 2, retrieve the triangle, and display messages showing its `Text`, `HTMLText`, and `RTFText` properties.

```
Dim MtbApp As New Mtb.Application
Dim MtbProj As Mtb.Project
Dim MtbTriangle As Mtb.Triangle

MtbApp.UserInterface.Visible = True
Set MtbProj = MtbApp.ActiveProject
MtbProj.ExecuteCommand "RANDOM 30 C1 - C3"
MtbProj.ExecuteCommand "CORR C1 C2 C3"

Set MtbTriangle = MtbProj.Commands(2).Outputs(2).Triangle

MsgBox "Triangle text: " & vbCrLf & _
    MtbTriangle.Text
MsgBox "Triangle HTMLText: " & vbCrLf & _
    MtbTriangle.HTMLText
MsgBox "Triangle RTFText: " & vbCrLf & _
    MtbTriangle.RTFText
```

Triangle property - HTMLText

Description
The contents of the `Triangle` object as HTML-formatted text.

Type
String

Range
Valid string

Access
Read-only

Triangle property - RTFText

Description
The contents of the `Triangle` object as RTF-formatted text.

Type
String

Range
Valid string

Access
Read-only

Triangle property - Text

Description
The contents of the `Triangle` object as unformatted text.

Type
String

Range
Valid string

Access
Read-only

Message object

Each `Message` object contains a single message generated by a Minitab `Command` on page 85 object.

Properties

Property	Description
HTMLText on page 118	The contents of the `Message` object as HTML-formatted text.
RTFText on page 118	The contents of the `Message` object as RTF-formatted text.
Text on page 118	The contents of the `Message` object as unformatted text.

Example

Create a new Minitab `Application` object, execute a command that generates a `Message` object as `Output` object number 3, retrieve the message, and display message boxes showing its `Text`, `HTMLText`, and `RTFText` properties.

```
Dim MtbApp As New Mtb.Application
Dim MtbProj As Mtb.Project
Dim MtbMsg As Mtb.Message

MtbApp.UserInterface.Visible = True
Set MtbProj = MtbApp.ActiveProject
MtbProj.ExecuteCommand "RANDOM 30 C1 - C3"
MtbProj.ExecuteCommand "CORR C1 C2 C3 C4"

Set MtbMsg = MtbProj.Commands(2).Outputs(3).Message

MsgBox "Message text: " & vbCrLf & _
    MtbMsg.Text
MsgBox "Message HTMLText: " & vbCrLf & _
    MtbMsg.HTMLText
MsgBox "Message RTFText: " & vbCrLf & _
    MtbMsg.RTFText
```

Message property - HTMLText

Description

The contents of the `Message` object as HTML-formatted text.

Type

String

Range

Valid string

Access

Read-only

Message property - RTFText

Description

The contents of the `Message` object as RTF-formatted text.

Type

String

Range

Valid string

Access

Read-only

Message property - Text

Description

The contents of the `Message` object as unformatted text.

Type
String

Range
Valid string

Access
Read-only