Notices


No part of this manual may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Agilent Technologies, Inc. as governed by United States and international copyright laws.

Edition
September, 2006
Document revision 3.2
Printed in USA
Agilent Technologies, Inc.
6612 Owens Dr.
Pleasanton, CA 94588-3334

Warranty
The material contained in this document is provided "as is," and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this manual and any information contained herein, including but not limited to the implied warranties of merchant-ability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

Technology Licenses
The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with such license.

Restricted Rights Legend
If software is for use in the performance of a U.S. Government prime contract or subcontract, Software is delivered and licensed as "Commercial computer software" as defined in DFAR 252.227-7014 (June 1995), or as a "commercial item" as defined in FAR 2.101(a) or as "Restricted computer software" as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Agilent Technologies’ standard commercial license terms, and non-DOD Departments and Agencies of the U.S. Government will receive no greater than Restricted Rights as defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June 1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.
# Table of Contents

**Introduction** ............................................................................................................. 17

**Object Hierarchy** ........................................................................................................ 18

**Automation Interface API’s** ...................................................................................... 19

- **Application Object** .................................................................................................. 20
  - Enterprise Property ..................................................................................................................... 20
  - Name Property .............................................................................................................................. 20
- **Enterprise Object** ........................................................................................................ 21
  - Instruments Property ..................................................................................................................... 21
  - LoginEnabled Property ..................................................................................................................... 21
  - Name Property .............................................................................................................................. 22
  - Projects Property .......................................................................................................................... 22
  - RefreshInstruments Method .......................................................................................................... 22
  - RefreshProjects Method ............................................................................................................... 23
  - UserListSource Property ............................................................................................................ 23
- **Instruments Object** .................................................................................................... 24
  - Count Property ............................................................................................................................. 24
  - Item Property ................................................................................................................................. 25
  - Name Property .............................................................................................................................. 25
- **Instrument Object** .................................................................................................... 26
  - AvailableLicenses Method .......................................................................................................... 26
  - Connect Method ........................................................................................................................... 28
  - ConnectOffline Method ................................................................................................................. 29
  - CurrentDataFile Property ............................................................................................................ 30
  - CurrentMethodFile Property ........................................................................................................ 30
  - CurrentSystemInfo Property ......................................................................................................... 30
  - DataAnalysis Property .................................................................................................................. 31
  - DataFileInfo Property .................................................................................................................... 31
  - DataSaveAs Method ....................................................................................................................... 31
  - Devices Property ........................................................................................................................... 32
  - Disconnect Method ....................................................................................................................... 32
  - EnterpriseLocation Property ........................................................................................................ 32
  - FractionColTraces Property ......................................................................................................... 33
  - GetConfigurationOptions Method ............................................................................................. 33
  - GetInstrumentCurrentState Method ........................................................................................... 34
  - GetLastRunCompletionState Method .......................................................................................... 34
  - ID Property ................................................................................................................................. 35
  - IsWindowLocked Method ............................................................................................................. 35
  - LoadDataFileWithOptions Method ............................................................................................. 36
  - LockWindow Method .................................................................................................................... 37
PropDilutFactor2 Property ...............................................................53
PropDilutFactor3 Property ...............................................................54
PropInitialVial Property ................................................................54
PropInitialVialText Property ..........................................................55
PropInjectionVolume Property .......................................................55
PropSTDAmount Property ..............................................................56
PropMethodName Property .............................................................56
PropMultFactor1 Property ..............................................................56
PropMultFactor2 Property ..............................................................57
PropMultFactor3 Property ..............................................................57
PropMultiplicationFactor Property .................................................58
PropPrintCalibrationReport Property ..............................................58
PropNumberOfReps Property ........................................................58
PropProject Property ......................................................................59
PropPrintCustomMethodReport Property .......................................59
PropSampleAmount Property .........................................................59
PropSampleDescription Property ..................................................60
PropSampleID Property ................................................................60
PropTowerToProcess Property .......................................................61
PropUserName Property ................................................................61
PropUserDomain Property .............................................................61
PropUserPassword Property ........................................................62
SetDefaultRunProperties Method ..................................................62
SubmitSingleRun Method ..............................................................63
SubmitPriorityRun Method ...........................................................64

**SequenceRun Object** .................................................................65
Analyze Method ..............................................................................65
Name Property ...............................................................................65
PropBracketingMode Property .......................................................65
PropEmailOnStart Property ...........................................................66
PropEmailOnStopOrError Property ...............................................66
PropEmailToAddress Property ......................................................67
PropPrintCalibrationReport Property .............................................67
PropPrintCustomMethodReport Property .......................................68
PropProcessingRunMode Property ................................................68
PropProject Property ......................................................................68
PropRangeMode Property .............................................................69
PropRangeSpecifier Property ........................................................69
PropSequenceName Property .........................................................70
PropTowerToProcess Property .......................................................70
PropUserDomain Property ............................................................70
PropUserName Property ...............................................................71
PropUserPassword Property ........................................................71
SetDefaultSequenceProperties Method ................................................................. 72
SubmitSequenceRun Method .................................................................................. 72

**DataAnalysis Object** .......................................................................................... 74
Name Property ........................................................................................................... 74
PropAverageCalibrationReplicates Property ......................................................... 74
PropCalibrationLevel Property ................................................................................ 74
PropClearAllCalibration Property ............................................................................ 75
PropClearCalibrationLevel Property ....................................................................... 75
PropClearCalibrationReplicates Property .............................................................. 76
PropDataName Property ........................................................................................... 76
PropDilutFactor1 Property ......................................................................................... 76
PropDilutFactor2 Property ......................................................................................... 77
PropDilutFactor3 Property ......................................................................................... 77
PropISTDAmount Property ......................................................................................... 78
PropMethodName Property ........................................................................................ 78
PropMultFactor1 Property ......................................................................................... 78
PropMultFactor2 Property ......................................................................................... 79
PropMultFactor3 Property ......................................................................................... 79
PropMultiplicationFactor Property .......................................................................... 80
PropPrintCalibrationReport Property ...................................................................... 80
PropPrintCustomMethodReport Property ................................................................ 80
PropSampleAmount Property .................................................................................... 81
PropSampleID Property ............................................................................................ 81
setDefaultAnalysisProperties Method .................................................................... 82
SubmitAnalysisEx Method ....................................................................................... 82
SubmitAnalysis Method ........................................................................................... 83

**CurrentSystemInfo Object** ................................................................................ 85
DataFileName Property .............................................................................................. 85
DilutorFactor1 Property ............................................................................................ 85
DilutorFactor2 Property ............................................................................................ 85
DilutorFactor3 Property ............................................................................................ 86
InstrumentName Property ......................................................................................... 86
ISTDAmount Property ............................................................................................... 86
MethodFileName Property ........................................................................................ 87
MultiplierFactor Property ......................................................................................... 87
MultiplierFactor1 Property ....................................................................................... 87
MultiplierFactor2 Property ....................................................................................... 88
MultiplierFactor3 Property ....................................................................................... 88
Name Property .......................................................................................................... 88
OEM_ID Property ....................................................................................................... 89
PretreatFileName Property ......................................................................................... 89
ProductName Property .............................................................................................. 89
SampleAmount Property .......................................................................................... 89
SampleID Property ................................................................. 90
SequenceFileName Property .................................................. 90
SoftwareBuild Property ......................................................... 90
SoftwareVersion Property ...................................................... 91
UserName Property .............................................................. 91

**DataFileInfo Object** ........................................................................................................ 92
AcquisitionDateTime Property ................................................ 92
AnalysisComputerName Property .......................................... 92
AnalysisDateTime Property ..................................................... 92

BCDValue Property ........................................................................ 93
Description Property ................................................................. 93
DilutorFactor1 Property .......................................................... 93
DilutorFactor2 Property .......................................................... 94
DilutorFactor3 Property .......................................................... 94
ESignatures Property ............................................................... 94

GetCustomParameter Method .................................................. 94
GetAuditTrailMode Method ...................................................... 95
InstrumentName Property ....................................................... 96
IsAuditTrailEnabled Method ..................................................... 96
IsElectronicallySigned Method ................................................. 96

ISTDAmount Property ............................................................... 97
MethodFileName Property ........................................................ 97
MultiplierFactor Property ........................................................ 97
MultiplierFactor1 Property ....................................................... 98
MultiplierFactor2 Property ....................................................... 98
MultiplierFactor3 Property ....................................................... 98
Name Property ........................................................................ 99

PDAFromWavelength Property ................................................ 99
PDAToWavelength Property ..................................................... 99
PDAWaveInterval Property ...................................................... 100
ReadOnly Property ........................................................................ 100
SampleAmount Property ........................................................ 100
SampleID Property ....................................................................... 101

SetAuditTrailMode Method ..................................................... 101
SoftwareVersion Property ....................................................... 101
UserName Property .............................................................. 102

Vial Property ........................................................................... 102
VialText Property .................................................................... 102
Volume Property ....................................................................... 103

**RunQueue Object** ........................................................................... 104
AbortMultipleRuns Method ..................................................... 104
Count Property ....................................................................... 105
Item Property ........................................................................ 106
Name Property .......................................................... 106
RefreshItemList Method .................................................. 107
Remove Method .......................................................... 107
RunQueueElem Object .................................................. 108
IsCurrentClient Method .................................................. 108
IsCurrentSession Method ............................................. 108
IsCurrentUser Method ................................................. 109
ItemDataName Property .............................................. 109
ItemDescription Property ........................................... 110
ItemID Property .......................................................... 110
ItemMethodName Property ........................................... 110
ItemPriorityFlag Property ............................................ 111
ItemSequenceName Property ........................................ 111
ItemStatus Property ..................................................... 111
ItemType Property ....................................................... 112
ItemUser Property ......................................................... 112
Name Property ........................................................... 113
Devices Object .......................................................... 114
Count Property ........................................................... 114
Item Property ............................................................. 114
Name Property ........................................................... 115
Device Object ........................................................... 116
Category Property ....................................................... 116
Channel_ID Property .................................................... 117
GetDeviceStatus Method .............................................. 117
InterfaceVersionNum Property ..................................... 117
Name Property ........................................................... 118
SendCommand Method ................................................. 118
SendCommandEx Method .............................................. 119
SendDirectControlCommand Method .......................... 119
VendorID Property ....................................................... 120
Projects Object ........................................................ 121
Count Property ........................................................ 121
Item Property ........................................................... 121
Name Property ........................................................... 122
Project Object ......................................................... 123
DataPath Property ....................................................... 123
Description Property .................................................. 123
MethodPath Property .................................................. 123
Name Property ........................................................... 124
ProjectPath Property ................................................... 124
SequencePath Property ............................................... 124
TemplatePath Property ............................................... 124
<table>
<thead>
<tr>
<th>Object Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traces Object</td>
<td>126</td>
</tr>
<tr>
<td>Count Property</td>
<td>126</td>
</tr>
<tr>
<td>Item Property</td>
<td>126</td>
</tr>
<tr>
<td>Name Property</td>
<td>127</td>
</tr>
<tr>
<td>Trace Object</td>
<td>128</td>
</tr>
<tr>
<td>ActualPointCount Property</td>
<td>128</td>
</tr>
<tr>
<td>ActualRuntime Property</td>
<td>128</td>
</tr>
<tr>
<td>ASTMNoise Property</td>
<td>128</td>
</tr>
<tr>
<td>BaseFrequency Property</td>
<td>129</td>
</tr>
<tr>
<td>DelayTime Property</td>
<td>129</td>
</tr>
<tr>
<td>DetectorName Property</td>
<td>129</td>
</tr>
<tr>
<td>Drift Property</td>
<td>130</td>
</tr>
<tr>
<td>ExpectedPointCount Property</td>
<td>130</td>
</tr>
<tr>
<td>ExpectedRuntime Property</td>
<td>130</td>
</tr>
<tr>
<td>GetTraceData Method</td>
<td>131</td>
</tr>
<tr>
<td>GetTracePoints Method</td>
<td>132</td>
</tr>
<tr>
<td>Groups Property</td>
<td>133</td>
</tr>
<tr>
<td>Name Property</td>
<td>133</td>
</tr>
<tr>
<td>Peaks Property</td>
<td>133</td>
</tr>
<tr>
<td>RefreshGroups Method</td>
<td>134</td>
</tr>
<tr>
<td>RefreshPeaks Method</td>
<td>134</td>
</tr>
<tr>
<td>RMSNoise Property</td>
<td>134</td>
</tr>
<tr>
<td>SamplingPeriod Property</td>
<td>135</td>
</tr>
<tr>
<td>SixSigmaNoise Property</td>
<td>135</td>
</tr>
<tr>
<td>StartTime Property</td>
<td>135</td>
</tr>
<tr>
<td>StopTime Property</td>
<td>136</td>
</tr>
<tr>
<td>TotalChanArea Property</td>
<td>136</td>
</tr>
<tr>
<td>TotalChanHeight Property</td>
<td>136</td>
</tr>
<tr>
<td>Type Property</td>
<td>136</td>
</tr>
<tr>
<td>UniformSampling Property</td>
<td>137</td>
</tr>
<tr>
<td>XAxisMultiplier Property</td>
<td>137</td>
</tr>
<tr>
<td>XAxisTitle Property</td>
<td>138</td>
</tr>
<tr>
<td>XAxisUnits Property</td>
<td>138</td>
</tr>
<tr>
<td>YAxisMultiplier Property</td>
<td>138</td>
</tr>
<tr>
<td>YAxisTitle Property</td>
<td>139</td>
</tr>
<tr>
<td>YAxisUnits Property</td>
<td>139</td>
</tr>
<tr>
<td>FractionCollTraces Object</td>
<td>140</td>
</tr>
<tr>
<td>Count Property</td>
<td>140</td>
</tr>
<tr>
<td>Item Property</td>
<td>140</td>
</tr>
<tr>
<td>Name Property</td>
<td>141</td>
</tr>
<tr>
<td>FractionCollTrace Object</td>
<td>142</td>
</tr>
<tr>
<td>FractionID Property</td>
<td>142</td>
</tr>
<tr>
<td>StartTime Property</td>
<td>142</td>
</tr>
</tbody>
</table>
StopTime Property ................................................................. 142
Vial_ID Property ...................................................................... 143
**Peaks Object** ...................................................................... 144
  Count Property ..................................................................... 144
  Item Property ....................................................................... 145
  Name Property ...................................................................... 145
**Peak Object** ..................................................................... 146
  AmplitudeBaselineStart Property ........................................ 146
  AmplitudeBaselineStop Property ........................................ 146
  ApexTime Property ................................................................ 146
  AreaPercent Property ....................................................... 147
  Asymmetry Property .......................................................... 147
  AsymmetryTenPercent Property ......................................... 147
  BackInflectionIndex Property ............................................. 148
  BaselineStartTime Property ................................................ 148
  BaselineStopTime Property ................................................ 149
  CapacityFactor Property .................................................... 149
  ConcentrationUnits Property .............................................. 149
  CorrectedArea Property ..................................................... 150
  Detected Property ............................................................. 150
  DetectedPeakNumber Property ........................................ 150
  DownInflectionBaselineX Property ...................................... 151
  DownInflectionBaselineY Property ...................................... 151
  DownSlopeSimilarity Property ........................................... 151
  ESTDConcentration Property ............................................. 152
  ESTDConcentrationValid Property ..................................... 152
  FrontInflectionIndex Property ............................................ 153
  GetCustomParameter Method ............................................. 153
  GroupNumber Property ..................................................... 153
  Height Property ................................................................... 154
  HeightPercent Property .................................................... 154
  InflectionX Property ......................................................... 155
  InflectionY Property ......................................................... 155
  IntegrationCodes Property ................................................ 155
  ISTDConcentration Property ............................................. 156
  ISTDConcentrationValid Property ..................................... 156
  ISTDPeakNumber Property ............................................... 156
  LambdaMax Property ........................................................ 157
  Name Property ................................................................... 157
  Named Property .................................................................. 157
  NamedPeakNumber Property ............................................. 158
  NORMConcentration Property .......................................... 158
  NORMConcentrationValid Property ................................... 159
Purity Property ................................................................. 159
QuantitationMethod Property ........................................... 159
ReferencePeakNumber Property ........................................ 160
RelativeRetentionTime Property .......................................... 160
Resolution Property .......................................................... 161
ResolutionUSP Property ..................................................... 161
ResolutionEMG Property ..................................................... 161
ResolutionDAB Property ..................................................... 162
ResolutionAOH Property ..................................................... 162
ResolutionJP Property ........................................................ 162
ResponseFactor Property ..................................................... 163
SimilarityIndex Property .................................................... 163
StandardPeakMultiplier Property ....................................... 163
StandardPeakNumber Property .......................................... 164
StopTime Property ............................................................. 164
StartTime Property ............................................................ 164
TheoreticalPlates Property .................................................. 165
TheoreticalPlatesPerMeter Property ................................. 165
Type Property ................................................................. 165
UpInflectionBaselineX Property ....................................... 166
UpInflectionBaselineY Property ....................................... 166
UpSlopeSimilarity Property .............................................. 166
UPWidth Property ........................................................... 167
Width Property ................................................................. 167
WidthAtFiftyPercent Property ........................................... 168
WidthAtFivePercent Property ............................................ 168
WidthAtTenPercent Property ............................................. 168
Groups Object ..................................................................... 170
Count Property ................................................................. 170
Item Property ................................................................. 170
Name Property ................................................................. 171
Group Object ...................................................................... 172
AreaPercent Property ....................................................... 172
ConcentrationUnits Property ........................................... 172
CorrectedArea Property ................................................... 172
ESTDConcentration Property .......................................... 173
ESTDConcentrationValid Property ................................... 173
GroupNumber Property ................................................... 173
Height Property ............................................................... 174
HeightPercent Property ................................................... 174
ISTDConcentration Property ............................................ 174
ISTDConcentrationValid Property ................................... 175
ISTDPeakNumber Property ................................................ 175
<table>
<thead>
<tr>
<th>Property Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name Property</td>
<td>176</td>
</tr>
<tr>
<td>NORMConcentration Property</td>
<td>176</td>
</tr>
<tr>
<td>NORMConcentrationValid Property</td>
<td>176</td>
</tr>
<tr>
<td>QuantitationMethod Property</td>
<td>177</td>
</tr>
<tr>
<td>ReferencePeakNumber Property</td>
<td>177</td>
</tr>
<tr>
<td>ResponseFactor Property</td>
<td>178</td>
</tr>
<tr>
<td>StandardPeakMultiplier Property</td>
<td>178</td>
</tr>
<tr>
<td>StandardPeakNumber Property</td>
<td>178</td>
</tr>
<tr>
<td><strong>MethodFileInfo Object</strong></td>
<td>180</td>
</tr>
<tr>
<td>AddNewCustomParam Method</td>
<td>180</td>
</tr>
<tr>
<td>AddPDA_MultiChrom Method</td>
<td>180</td>
</tr>
<tr>
<td>AfterAnalysisAdditionalParams Property</td>
<td>181</td>
</tr>
<tr>
<td>AfterAnalysisFileName Property</td>
<td>181</td>
</tr>
<tr>
<td>AfterExportAdditionalParams Property</td>
<td>181</td>
</tr>
<tr>
<td>AfterExportFileName Property</td>
<td>182</td>
</tr>
<tr>
<td>BaselineFileName Property</td>
<td>182</td>
</tr>
<tr>
<td>BeforeAnalysisAdditionalParams Property</td>
<td>182</td>
</tr>
<tr>
<td>BeforeAnalysisFileName Property</td>
<td>182</td>
</tr>
<tr>
<td>BeforeRunAdditionalParams Property</td>
<td>183</td>
</tr>
<tr>
<td>BeforeRunFileName Property</td>
<td>183</td>
</tr>
<tr>
<td>ChangePDA_MultiChrom Method</td>
<td>183</td>
</tr>
<tr>
<td>ClearAllMethodProgramSteps Method</td>
<td>184</td>
</tr>
<tr>
<td>DeleteCustomParam Method</td>
<td>184</td>
</tr>
<tr>
<td>ExportEnabled Property</td>
<td>185</td>
</tr>
<tr>
<td>GetAuditTrailMode Method</td>
<td>185</td>
</tr>
<tr>
<td>GetChannelIDsToSubtract Method</td>
<td>186</td>
</tr>
<tr>
<td>GetMethodFirstProgramValue Method</td>
<td>186</td>
</tr>
<tr>
<td>GetMethodItemValue Method</td>
<td>187</td>
</tr>
<tr>
<td>GetMethodNextProgramValue Method</td>
<td>187</td>
</tr>
<tr>
<td>GetPDA_BOOLData Method</td>
<td>188</td>
</tr>
<tr>
<td>GetPDA_INT32Data Method</td>
<td>189</td>
</tr>
<tr>
<td>GetPDA_MultiChrom Method</td>
<td>190</td>
</tr>
<tr>
<td>GetPDA_MultiChromTableSize Method</td>
<td>190</td>
</tr>
<tr>
<td>GetValidValueRange Method</td>
<td>191</td>
</tr>
<tr>
<td>IsAuditTrailEnabled Method</td>
<td>191</td>
</tr>
<tr>
<td>Method Item IDs</td>
<td>192</td>
</tr>
<tr>
<td>MethodCustomParams Property</td>
<td>193</td>
</tr>
<tr>
<td>MethodPropAnalysisPeriod Property</td>
<td>193</td>
</tr>
<tr>
<td>MethodPropAnalyzeAfterAcq Property</td>
<td>193</td>
</tr>
<tr>
<td>MethodPropAnalyzeDuringAcq Property</td>
<td>194</td>
</tr>
<tr>
<td>MethodPropAutoAvgReplicates Property</td>
<td>194</td>
</tr>
<tr>
<td>MethodPropEnableDataFileCompress Property</td>
<td>194</td>
</tr>
<tr>
<td>MethodPropertiesDescription Property</td>
<td>194</td>
</tr>
</tbody>
</table>
MethodPropNumRepsInRollingAvg Property ................................................................. 195
MethodPropResponseFactorDef Property .................................................................. 195
Name Property ........................................................................................................... 195
RefreshMethodCustomParams Method ...................................................................... 196
RemoveAllPDA_MultiChroms Method ...................................................................... 196
RemovePDA_MultiChrom Method ............................................................................. 196
SetAfterAnalysisData Method .................................................................................. 197
SetAfterExportData Method ..................................................................................... 197
SetAuditTrailMode Method ....................................................................................... 198
SetBaselineData Method .......................................................................................... 198
SetBeforeAnalysisData Method ................................................................................ 199
SetBeforeRunData Method ....................................................................................... 199
SetChannelIDsToSubtract Method .......................................................................... 200
SetMethodItemValue Method .................................................................................... 200
SetMethodOptionsData Method ................................................................................ 201
SetMethodProgramValue Method ............................................................................. 201
SetPDA_BOOLData Method ...................................................................................... 202
SetPDA_INT32Data Method ..................................................................................... 202
UseAfterAnalysisProgram Property ........................................................................ 203
UseAfterExportProgram Property ......................................................................... 204
UseBaselineFile Property ....................................................................................... 204
UseBeforeAnalysisProgram Property .................................................................... 204
UseBeforeRunProgram Property ........................................................................... 204
MethodCustomParams Object ................................................................................. 206
  Count Property ........................................................................................................ 206
  Item Property .......................................................................................................... 206
  Name Property ........................................................................................................ 207
MethodCustomParam Object ...................................................................................... 208
  AdditionalParams Property .................................................................................... 208
  InUse Property ........................................................................................................ 208
  ParameterID Property ............................................................................................ 208
  ParameterName Property ....................................................................................... 209
  ParamType Property .............................................................................................. 209
  ReturnType Property .............................................................................................. 210
  SetMethodCustomParamData Method .................................................................. 210
  SourceFile Property .............................................................................................. 211
MethodChannels Object ............................................................................................. 212
  Count Property ...................................................................................................... 212
  Item Property .......................................................................................................... 213
  Name Property ........................................................................................................ 213
MethodChannel Object ............................................................................................... 214
  CalcPerformanceParams Property .................................................................... 214
  CalcUsingAOH Property ....................................................................................... 214
CalcUsingDAB_BP_EP_ASTM Property ................................................................. 214
CalcUsingEMG Property .......................................................... 215
CalcUsingJP Property ................................................................. 215
CalcUsingUSP Property ................................................................. 215
Channel_ID Property ................................................................. 216
ColumnDescription Property ...................................................... 216
ColumnInstallationDate Property .................................................. 216
ColumnLength Property ............................................................... 217
ColumnSerialNum Property ......................................................... 217
ParticleDiameter Property ............................................................ 217
SetMethodPerformanceParamsData Method ................................... 218
UnretainedPeakTime Property ....................................................... 218
MethodTraces Object ........................................................................ 220
Count Property ........................................................................ 220
Item Property ........................................................................ 220
Name Property ........................................................................ 221
MethodTrace Object ........................................................................ 222
AddNewNamedPeak Method ........................................................ 222
Channel_ID Property ................................................................. 222
DeleteNamedPeak Method ........................................................... 222
ExportPath Property ...................................................................... 223
IntegrationEvents Property ......................................................... 223
NamedPeaks Property ................................................................. 223
RefreshIntegrationEvents Method ............................................... 224
RefreshNamedPeaks Method ........................................................ 224
TraceID Property ......................................................................... 225
TraceIDEx Property ...................................................................... 225
TraceName Property ...................................................................... 225
IntegrationEvents Object ............................................................. 227
Count Property ........................................................................ 227
Item Property ........................................................................ 227
Name Property ........................................................................ 228
IntegrationEvent Object .............................................................. 229
EventType Property ...................................................................... 229
EventUsed Property ..................................................................... 230
StartTime Property ...................................................................... 230
StopTime Property ...................................................................... 230
Value Property ........................................................................ 231
NamedPeaks Object ................................................................. 232
Count Property ........................................................................ 232
Item Property ........................................................................ 232
Name Property ........................................................................ 233
NamedPeak Object ...................................................................... 234
SetNamedPeakData Method ................................................................. 248
SimThreshold Property ................................................................. 249
SpectrumFileName Property ............................................................ 250
Spike1Amount Property ................................................................. 250
Spike2Amount Property ................................................................. 250
StdID Property .............................................................................. 251
StdMult Property .......................................................................... 251
WeightingMethod Property ............................................................ 251
ESignatures Object ........................................................................ 253
Count Property .............................................................................. 253
Item Property ................................................................................ 253
Name Property .............................................................................. 254
ESignature Object ........................................................................ 255
Comment Property ........................................................................ 255
DateTimeStamp Property ............................................................... 255
Reason Property ........................................................................... 255
Role Property ................................................................................. 256
UserName Property ........................................................................ 256
Introduction

This document describes the API available for automating the EZChrom Elite data system. Using this interface, it is possible to create simplified user interfaces for the data system.

Note: In order to use the automation features described in this document, you must have the Automation Option installed on your system.

The control of Elite is accomplished through Automation (formerly OLE Automation). The objects are implemented using dual interfaces; meaning they may be driven by either early bound (VTBL) or late bound (IDispatch) calls. See the documentation for the development tool you are using to determine how early or late binding is accomplished with that tool. When possible, early binding should be used as this is much more efficient and your applications will execute faster.

The Automation interface is implemented in the "EZChromAutomation.exe" file and the type library is defined in "EZChromAutomation.tlb". For C/C++ programmers, the interface declarations and Type IDs are in "EZChromAutomation_i.c" and "EZChromAutomation_i.h". The programmatic identifier of the automation object is "EZChrom.Application". Once an object of this kind is created, then the functions described in this document may be used.

All examples in this document are given in Visual Basic. However, any other language that supports Automation should be able to accomplish the same tasks.

Note to Visual Basic programmers: Because of the difference between C and Visual Basic Boolean values, only test returned values for False. A returned value of False is a False in Visual Basic. Any other value should be considered True. Do not test return values for True.

Note: This document is provided for information only. Customers are not entitled to technical support for programs or applications written using this document. Customers can obtain technical support through a custom support agreement with Agilent Technologies Inc.
Automation Interface API’s

The following documentation describes the functions and properties that are available through the Elite Automation interface. The required inputs and outputs are described.
Application Object

This is the top-level object in the automation interface. This object is created when the initial connection to this automation server is established. For an overview of all objects, see the Object Hierarchy.

Example
This example creates an Application Object using early binding:

```vba
Dim EZAppObject As EZChromAutomation.Application
Set EZAppObject = CreateObject("EZChrom.Application")
```

Enterprise Property

Returns the Enterprise object. This is a member of the Application object.

Type
Read Only Object

Example
This example determines whether or not Instrument Login and Project Management is enabled, and displays a message box to the user:

```vba
bLoginEnabled = Application.Enterprise.LoginEnabled
If (bLoginEnabled = True) Then
    MsgBox ("Login is enabled")
Else
    MsgBox ("Login is NOT enabled")
End If
```

Name Property

Returns the name of this object. This is a member of the Application object.

Type
Read Only String

Example
This example displays a message box with the name of this object to the user:

```vba
strMsg = "The name of the Application object is " & _
          Application.Name
MsgBox (strMsg)
```
**Enterprise Object**

This object contains information about the users, instruments, and projects that are used by the EZChrom Elite application. For an overview of all objects, see the Object Hierarchy.

**Instruments Property**

Returns the `Instruments` object. This is a member of the `Enterprise` object.

**Type**
Read Only Object

**Example**
This example displays the name of each configured instrument to the user:

```vba
For Each Instrument In Enterprise.Instruments
    MsgBox ("Instrument Name: " & Instrument.Name)
Next
```

**LoginEnabled Property**

Returns TRUE or FALSE to indicate if instrument login and project management is enabled. This is a member of the `Enterprise` object.

**Type**
Read Only Boolean

**Example**
This example determines whether or not Instrument Login and Project Management is enabled, and displays a message box to the user:

```vba
bLoginEnabled = Enterprise.LoginEnabled
If (bLoginEnabled = True) Then
    MsgBox ("Login is enabled")
Else
    MsgBox ("Login is NOT enabled")
End If
```
**Name Property**

Returns the name of this object. This is a member of the Enterprise object.

**Type**
Read Only String

**Example**
This example displays a message box with the name of this object to the user:

```vbnet
strMsg = "The name of the Enterprise object is " & _
         Enterprise.Name
MsgBox (strMsg)
```

**Projects Property**

Returns the Projects object. This is a member of the Enterprise object.

**Type**
Read Only Object

**Example**
This example displays the name of each project to the user:

```vbnet
For Each Project In Enterprise.Projects
    MsgBox ("Project Name: " & Project.Name)
Next
```

**RefreshInstruments Method**

Refreshes the Instruments object, which reloads the instrument list. This is a member of the Enterprise object.

**Returns**
Void

**Syntax**
RefreshInstruments()

**Example**
This example displays the most recent list of Instruments:

```vbnet
Enterprise.RefreshInstruments
For Each Instrument In Enterprise.Instruments
    MsgBox Instrument.Name
Next
```
**RefreshProjects Method**

Refreshes the *Projects* object, which reloads the project list. This is a member of the *Enterprise* object.

**Returns**
Void

**Syntax**
```
RefreshProjects()
```

**Example**
This example displays the most recent list of Projects:
```
Enterprise.RefreshProjects
For Each Project In Enterprise.Projects
    MsgBox Project.Name
Next
```

**UserListSource Property**

Returns a flag to indicate if the user lists come from the data system or a domain controller. This is a member of the *Enterprise* object.

The value returned is one of the following *EZUserSourceFlags* constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ezUserDataSystem</td>
<td>The user lists are generated and maintained by the EZChrom Elite Data System.</td>
</tr>
<tr>
<td>ezUserDomainController</td>
<td>The user lists are obtained from a Windows NT Domain Controller.</td>
</tr>
</tbody>
</table>

**Type**
Read Only Long (4 byte integer value)

**Example**
This example determines whether or not Instrument Login and Project Management is enabled, and displays a message box to the user:
```
If (Enterprise.UserListSource = ezUserDataSystem) Then
    MsgBox ("User lists are generated from the data system")
Else
    MsgBox ("User lists are generated from the NT Domain Controller")
End If
```
Instruments Object

This object contains a collection of all the Instrument objects that are currently available. Applications can enumerate through the collection of instruments by using For … Next statements or by using For … Each statements. For an overview of all objects, see the Object Hierarchy.

Example
This example displays the status of each instrument in the collection using For … Each statements:

    For Each Instrument In Instruments
        MsgBox ("Instrument Status: " & Instrument.Status)
    Next

This example displays the status of each instrument in the collection using For … Next statements:

    nCount = Instruments.Count
    For nIndex = 0 To nCount - 1
        MsgBox ("Instrument Status: " & Instruments(nIndex).Status)
    Next nIndex

Count Property

Returns the number of instruments available. This is a member of the Instruments object.

Type
Read Only Long (4 byte integer value)

Example
This example displays the status of each instrument to the user:

    nCount = Instruments.Count
    For nIndex = 0 To nCount - 1
        MsgBox ("Instrument Status: " & Instruments(nIndex).Status)
    Next nIndex
**Item Property**

Returns the Instrument object at the given indexed location. This is a member of the Instruments object.

**Type**
Read Only Object

**Syntax**
Item(Index)

**Index**
Required Long (4 byte integer value). This is a zero based index that specifies the Instrument Object to be returned.

**Example**
This example displays the status of each instrument in the collection:

```vbnet
nCount = Instruments.Count
For nIndex = 0 To nCount - 1
    MsgBox ("Instrument Status: " & Instruments.Item(nIndex).Status)
Next nIndex
```

**Remarks**
Item is the default member of this collection, so the following two lines of code are equivalent:

```vbnet
Instruments.Item(1)
Instruments(1)
```

**Name Property**

Returns the name of this object. This is a member of the Instruments object.

**Type**
Read Only String

**Example**
This example displays a message box with the name of this object to the user:

```vbnet
strMsg = "The name of the Instruments object is " & _
         Instruments.Name
MsgBox (strMsg)
```
**Instrument Object**

This object represents an Instrument that has been configured for use. The `Instruments` collection object contains all of the Instrument objects that are currently available. For an overview of all objects, see the [Object Hierarchy](#).

**NOTE:** All methods and properties of this object must be used with a connected instrument or a run time error will occur, with the exception of the following:

- Connect Method
- ConnectOffline Method
- Disconnect Method
- EnterpriseLocation Property
- ID Property
- Name Property
- Status Property

**AvailableLicenses Method**

Requests the number of available licenses for a specific key ID. This is a member of the `Instrument` object.

**Returns**

Long (4 byte integer value)

**Syntax**

`AvailableLicenses(KeyID)`

**KeyID**

Required Long (4 byte integer value). This key ID is required to request how many licenses are available for it.

The key ID required needs to be one of the following constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OEM HPLC</td>
</tr>
<tr>
<td>3</td>
<td>Client count</td>
</tr>
<tr>
<td>4</td>
<td>OEM CE</td>
</tr>
<tr>
<td>7</td>
<td>Super Compare</td>
</tr>
<tr>
<td>8</td>
<td>PDA</td>
</tr>
<tr>
<td>9</td>
<td>Pump</td>
</tr>
<tr>
<td>11</td>
<td>OEM GC</td>
</tr>
<tr>
<td>12</td>
<td>Instrument Options</td>
</tr>
<tr>
<td>13</td>
<td>OEM Autosampler</td>
</tr>
<tr>
<td>14</td>
<td>BTU</td>
</tr>
<tr>
<td>15</td>
<td>HP Autosampler</td>
</tr>
<tr>
<td>17</td>
<td>Offline Options</td>
</tr>
<tr>
<td>18</td>
<td>SEC</td>
</tr>
<tr>
<td>19</td>
<td>System Suitability</td>
</tr>
</tbody>
</table>
Example
This example requests the available licenses for the key ID client count, then the user is notified with a message box:
Dim availLicenses as Long
availLicenses = Instrument.AvailableLicenses(3)
MsgBox("The number of available client count " & " licenses is " & availLicenses)

**Connect Method**

Creates an active connection with the current instrument. Optionally, this will log the specified user into the instrument for the specified project. This is a member of the `Instrument` object.

The value returned is one of the following **EZConnectionFlags** constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ezConnectOk</td>
<td>The connection to the instrument was performed successfully.</td>
</tr>
<tr>
<td>ezConnectAlreadyConnected</td>
<td>A connection has already been made to this instrument from this client.</td>
</tr>
<tr>
<td>ezConnectAccountInvalid</td>
<td>One or more of the &quot;User Name&quot;, &quot;Password&quot;, or &quot;Domain&quot; is invalid.</td>
</tr>
<tr>
<td>ezConnectNoInstrumentPrivileges</td>
<td>The specified user does not have access privileges to connect to this instrument.</td>
</tr>
<tr>
<td>ezConnectNoProjectPrivileges</td>
<td>The specified user does not have access privileges to the specified project.</td>
</tr>
<tr>
<td>ezConnectGeneralFailure</td>
<td>A general error occurred while connecting to this instrument.</td>
</tr>
<tr>
<td>ezConnectInsufficientLicenses</td>
<td>Insufficient licenses are available to make the connection to the instrument.</td>
</tr>
<tr>
<td>ezConnectDemoModeError</td>
<td>It is not possible to connect to this instrument while the system is in Demo mode.</td>
</tr>
</tbody>
</table>

**Returns**

Long (4 byte integer value)

**Syntax**

`Connect(UserName, Password, Domain, Project)`

- **UserName** Optional String. This is the name of the user to be connected to this instrument. If this value is not specified a default value of "" will be used.
- **Password** Optional String. This is the password of the user to be connected to this instrument. If this value is not specified a default value of "" will be used.
- **Domain** Optional String. This is the domain of the user to be connected to this instrument. If this value is not specified a default value of "" will be used.
- **Project** Optional String. This is the name of the project to be used with the connected instrument. If this value is not specified a default value of "Default" will be used. This allows the "Default" project to be used without having to specify it.

**Example**

This example performs a connection to the current instrument for "User A" on the "TEST" domain. If an error occurs, then the user is notified with a message box:

```vba
Dim retval as EZChromAutomation.EZConnectionFlags
```
retval = Instrument.Connect("User A", "xyz", "TEST")
If retval <> ezConnectOk Then
    MsgBox ("Unable to connect with instrument!")
End If

ConnectOffline Method

Creates an offline connection with the current instrument. Optionally, this will log the specified user into the instrument for the specified project. This is a member of the Instrument object.

The value returned is one of the following EZConnectionFlags constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ezConnectOk</td>
<td>The connection to the instrument was performed successfully.</td>
</tr>
<tr>
<td>ezConnectAlreadyConnected</td>
<td>A connection has already been made to this instrument from this client.</td>
</tr>
<tr>
<td>ezConnectAccountInvalid</td>
<td>One or more of the &quot;User Name&quot;, &quot;Password&quot;, or &quot;Domain&quot; is invalid.</td>
</tr>
<tr>
<td>ezConnectNoInstrumentPrivileges</td>
<td>The specified user does not have access privileges to connect to this instrument.</td>
</tr>
<tr>
<td>ezConnectNoProjectPrivileges</td>
<td>The specified user does not have access privileges to the specified project.</td>
</tr>
<tr>
<td>ezConnectGeneralFailure</td>
<td>A general error occurred while connecting to this instrument.</td>
</tr>
<tr>
<td>ezConnectInsufficientLicenses</td>
<td>Insufficient licenses are available to make the connection to the instrument.</td>
</tr>
<tr>
<td>ezConnectDemoModeError</td>
<td>It is not possible to connect to this instrument while the system is in Demo mode.</td>
</tr>
</tbody>
</table>

Returns
Long (4 byte integer value)

Syntax
Connect(UserName, Password, Domain, Project)

UserName  Optional String. This is the name of the user to be connected to this instrument. If this value is not specified a default value of "" will be used.
Password   Optional String. This is the password of the user to be connected to this instrument. If this value is not specified a default value of "" will be used.
Domain     Optional String. This is the domain of the user to be connected to this instrument. If this value is not specified a default value of "" will be used.
Project    Optional String. This is the name of the project to be used with the connected instrument. If this value is not specified a default value of "Default" will be used. This allows the "Default" project to be used without having to specify it.

Example
This example performs an offline connection to the current instrument for "User A" on the "TEST" domain. If an error occurs, then the user is notified with a message box:

Dim retval as EZChromAutomation.EZConnectionFlags
retval = Instrument.ConnectOffline("User A", "xyz", "TEST")
If retval <> ezConnectOk Then
    MsgBox ("Unable to connect with instrument!")
End If

**CurrentDataFile Property**

Sets or returns the data file that is currently loaded by this instrument. This is a member of the 
**Instrument** object.

**Type**
Read/Write String

**Example**
This example makes the current instrument load the specified data file:

```
Instrument.CurrentDataFile = "C:\EZChrom Elite\Data001.dat"
```

**CurrentMethodFile Property**

Sets or returns the method file that is currently loaded by this instrument. This is a member of the 
**Instrument** object.

**Type**
Read/Write String

**Example**
This example makes the current instrument load the specified method file:

```
Instrument.CurrentMethodFile = "C:\EZChrom Elite\Method001.met"
```

**CurrentSystemInfo Property**

Returns the **CurrentSystemInfo** object. This is a member of the **Instrument** object.

**Type**
Read Only Object

**Example**
This example determines the ISTD amount from the last analysis, and displays the result in a 
message box to the user:

```
fISTDAmount = Instrument.CurrentSystemInfo.ISTDAmount
MsgBox ("ISTD amount: " & fISTDAmount)
```
**DataAnalysis Property**

Returns the **DataAnalysis** object. This is a member of the **Instrument** object.

**Type**
Read Only Object

**Example**
This example performs an analysis using the specified method and data file:

```vba
Instrument.DataAnalysis.PropMethodName = "C:\EZChrom Elite\Method001.met"
Instrument.DataAnalysis.PropDataName = "C:\EZChrom Elite\Data001.dat"
Instrument.DataAnalysis.SubmitAnalysis( _
   ezAnalysisSpecifiedOptions)
```

**DataFileInfo Property**

Returns the **DataFileInfo** object. This is a member of the **Instrument** object.

**Type**
Read Only Object

**Example**
This example determines the name of the method that was used to acquire the data, and displays
the result in a message box to the user:

```vba
strMethodName = Instrument.DataFileInfo.MethodFileName
MsgBox ("Method name: " & strMethodName)
```

**DataSaveAs Method**

Saves the current data along with the current method in the same file. This command will succeed
only when the current data file is not already a 32-bit Elite data format file (for example, a 16-bit file
or a converted file). This is a member of the **Instrument** object.

**Returns**
Boolean

**Syntax**
DataSaveAs(strDataFileName)

**Example**
This example loads previously exported ASCII data and then saves it as a 32-bit data file.

```vba
Dim strFileName As String
Dim lDataOpenFlag As Long
Dim lDataResultFlag As Long
```
strFileName = "C:\EZChrom Elite\datafile.asc"
lDataOpenFlag = ezOpenDataOnly
lDataResultFlag = ezDataResultsNone

Instrument.LoadDataFileWithOptions strFileName, lDataOpenFlag, _
    lDataResultFlag

Instrument.DataSaveAs("datafile.dat")

**Devices Property**

Returns the **Devices** object. This is a member of the **Instrument** object.

**Type**
Read Only Object

**Example**
This example displays the name of each instrument device to the user:

```vba
For Each Device In Instrument.Devices
    MsgBox ("Device Name: " & Device.Name)
Next
```

**Disconnect Method**

Disconnects an active connection with the current instrument. This method will be called automatically if the client application exits without performing this call. This is a member of the **Instrument** object.

**Returns**
Void

**Syntax**
Disconnect()

**Example**
This example disconnects a previous connection that was made to the current instrument:

```vba
Instrument.Disconnect
```

**EnterpriseLocation Property**

Returns the location of this instrument in the enterprise hierarchy. This is a member of the **Instrument** object.

**Type**
Read Only String

**Example**
This example displays the enterprise location of the current instrument to the user:

MsgBox ("Enterprise location of this instrument: " & _
Instrument.EnterpriseLocation)

**FractionCollTraces Property**

Returns the `FractionCollTraces` object. This is a member of the `Instrument` object.

**Type**
Read Only Object

**Example**
This example displays the fraction ID of each fraction collection trace to the user:

```vba
For Each FractionCollTrace In Instrument.FractionCollTraces
    MsgBox ("Fraction ID: " & FractionCollTrace.FractionID)
Next
```

**GetConfigurationOptions Method**

Puts the Configuration Options data into the variant. This is a member of the `Instrument` object.

**Returns**
Void

**Syntax**
`GetConfigurationOptions(vConfigurationOptions)`

**vConfigurationOptions**
Variant. This is a variant that will receive a list of longs (4 byte integer values). They will be listed in ascending order. The following table shows the meaning of the numbers currently used. If an option is enabled it's number will be in the list. A number not in the list means that the option is NOT enabled. If no options are configured, the returned variant type is 0.

<table>
<thead>
<tr>
<th>Option Number</th>
<th>Option Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>System Suitability</td>
</tr>
<tr>
<td>1</td>
<td>SEC</td>
</tr>
<tr>
<td>2</td>
<td>PDA</td>
</tr>
<tr>
<td>3</td>
<td>Chromatopack</td>
</tr>
<tr>
<td>4</td>
<td>CE</td>
</tr>
<tr>
<td>5</td>
<td>Caesar</td>
</tr>
<tr>
<td>6</td>
<td>QA</td>
</tr>
<tr>
<td>7</td>
<td>System Check</td>
</tr>
<tr>
<td>8</td>
<td>RGA</td>
</tr>
<tr>
<td>9</td>
<td>Baseline Check</td>
</tr>
</tbody>
</table>

**Example**
This example gets the Configuration Options:
GetInstrumentCurrentState Method

Get the current instrument state and sample ID. This is a member of the Instrument object. Unlike Instrument.Status, Instrument.GetInstrumentCurrentState should only be invoked after the instrument is launched.

Returns
Read Only Long (4 byte integer value).

The value returned is one of the following EZInstrumentStates constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ezIdle</td>
<td>The instrument is in the Idle state.</td>
</tr>
<tr>
<td>ezInitializing</td>
<td>The instrument is initializing.</td>
</tr>
<tr>
<td>ezDownloadingMethod</td>
<td>The instrument is downloading the current method.</td>
</tr>
<tr>
<td>ezEquilibrating</td>
<td>The instrument is equilibrating.</td>
</tr>
<tr>
<td>ezWaitingForTrigger</td>
<td>The instrument is waiting for a trigger.</td>
</tr>
<tr>
<td>ezRunning</td>
<td>The instrument is running (acquiring data).</td>
</tr>
<tr>
<td>ezCompletingRun</td>
<td>The instrument is either executing post run tasks or it is aborting a run.</td>
</tr>
<tr>
<td>ezSystemCheck</td>
<td>The instrument is executing a system check.</td>
</tr>
<tr>
<td>ezBaselineCheck</td>
<td>The instrument is executing a baseline check.</td>
</tr>
<tr>
<td>ezUnknown</td>
<td>The instrument is in a state that the automation software does not know about. Please call tech support if this state is returned.</td>
</tr>
</tbody>
</table>

Syntax
GetInstrumentCurrentState(SampleID)

SampleID Read Only String. This is a string that will receive the current sample ID if there is one.

Example
This example gets the instrument state and sample ID:

```
Dim lngInstState As Long
Dim strSampleID As String
```

GetLastRunCompletionState Method

Gets the last completed run's ending status and sample ID. This is a member of the Instrument object.

Returns
Read Only Long (4 byte integer value).

The value returned is one of the following **EZLastCompletedRunStates** constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ezNoLastRun</td>
<td>There is no completed last run.</td>
</tr>
<tr>
<td>ezLastRunCompleteSuccess</td>
<td>The last run completed successfully.</td>
</tr>
<tr>
<td>ezLastRunUserAborted</td>
<td>The last run was aborted by the user.</td>
</tr>
<tr>
<td>ezLastRunHardwareAborted</td>
<td>The last run was aborted due to a hardware failure.</td>
</tr>
</tbody>
</table>

**Syntax**
GetLastRunCompletionState(SampleID)

**Example**
This example gets the last completed run’s ending state and sample ID:

```vba
Dim lngLastRun As Long
Dim strSampleID As String
lngLastRun = Instrument.GetLastRunCompletionState(strSampleID)
```

**ID Property**

Returns the internal instrument ID of this instrument. This is the number displayed to the user through the properties dialog of an instrument in the Main Menu of the EZChrom Elite application. This is a member of the **Instrument** object.

**Type**
Read Only Long (4 byte integer value)

**Example**
This example obtains the instrument ID of the current instrument and displays a message box for the user:

```vba
nInstrumentID = Instrument.ID
MsgBox ("Instrument ID: " & nInstrumentID)
```

**IsWindowLocked Method**

Returns TRUE when the current instrument window is locked, FALSE otherwise. This method is only valid for stand-alone systems. In a client/server environment, this method is not valid. This is a member of the **Instrument** object.

**Returns**
Boolean
Example
This example determines the lock state of the instrument window.

Dim bLocked as Boolean
bLocked = Instrument.IsWindowLocked

**LoadDataFileWithOptions Method**

Loads the specified data file into the current instrument, using the specified options. This is a member of the Instrument object.

Returns
Void

Syntax
LoadDataFileWithOptions(DataFileName, DataOpenFlags, DataResultFlags)

DataFileName String. This is the name of the data file to be loaded.

DataOpenFlags Optional Long (4 byte integer value). This is a flag, from the EZDataFileOpenFlags enumeration, that specifies any additional files that are to be opened with the data file. If this value is not specified a default value of ezOpenDataOnly will be used.

DataResultFlags Optional Long (4 byte integer value). This is a flag, from the EZDataResultFlags enumeration, that specifies what sort of results/method should be loaded with the specified data file. If this value is not specified a default value of ezDataResultsNone will be used.

The EZDataFileOpenFlags enumeration contains the following constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ezOpenDataOnly</td>
<td>Open the specified data file only.</td>
</tr>
<tr>
<td>ezOpenWithMethod</td>
<td>Open the specified data file with its embedded method.</td>
</tr>
<tr>
<td>ezOpenWithConfiguration</td>
<td>Open the specified data file with its embedded configuration.</td>
</tr>
</tbody>
</table>

The EZDataResultFlags enumeration contains the following constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ezDataResultsNone</td>
<td>Open the specified data file only. Do not load any results or method file.</td>
</tr>
<tr>
<td>EzDataResultsLast</td>
<td>Open the specified data file and load the last results.  Also, if the embedded method is being loaded then load the method file that was used to analyze the last results.</td>
</tr>
<tr>
<td>EzDataResultsOriginal</td>
<td>Open the specified data file and load the original results.  Also, if the embedded method is being loaded then load the method file that was used to initially collect the data file.</td>
</tr>
</tbody>
</table>

Example
This example loads a data file with the original results:

```vba
Instrument.LoadDataFileWithOptions(  
    "C:\EZChrom Elite\Data001.dat", ezOpenDataOnly, _  
    ezDataResultsOriginal)
```

**LockWindow Method**

Locks the instrument window. This method is only valid for stand-alone systems. In a client/server environment, this method is not valid. Returns TRUE if the attempt to lock the window was successful. This is a member of the `Instrument` object.

**Returns**

Boolean

**Example**

This example locks the instrument window.

```vba
Call Instrument.LockWindow
```

**MethodChannels Property**

Returns the `MethodChannels` object. This is a member of the `Instrument` object.

**Type**

Read Only Object

**Example**

This example displays the Column Serial Number of each channel to the user:

```vba
For Each MethodChannel In Instrument.MethodChannels
    MsgBox ("Column Serial #: " & _
            MethodChannel.ColumnSerialNum)
Next
```

**MethodDownload Method**

Downloads the current method to the instrument. This is a member of the `Instrument` object.

**Returns**

Boolean

**NOTE:** The return value only indicates whether the command was submitted for execution or not and does NOT indicate whether the method was actually downloaded to the instrument.

**Syntax**

```vba
MethodDownload()
```

**Example**
This example downloads the current method to the instrument.

\[
b\text{MethodDownloadCmdSent} = \text{Instrument.MethodDownload}
\]

**MethodFileInfo Property**

Returns the `MethodFileInfo` object. This is a member of the `Instrument` object.

**Type**
Read Only Object

**Example**
This example gets the method properties description, and displays the result in a message box to the user:

\[
\text{strMethodPropDesc} = \\
\text{Instrument.MethodFileInfo.MethodPropertiesDescription} \\
\text{MsgBox ("Method Properties Description: " & strMethodPropDesc)}
\]

**MethodLoad Method**

Opens the specified method file. This is a member of the `Instrument` object.

**Returns**
Boolean

**Syntax**
```
MethodLoad(MethodFileName)
```

**MethodFileName**  String. This is the name of the method file to be opened.

**Example**
This example opens a method file, thereby making it the current method:

\[
b\text{MethodWasOpened} = \text{Instrument.MethodLoad(} \\
\text{"C:\EZChrom Elite\Method001.met")}
\]

**MethodModified Property**

Returns whether the current method has been modified or not. This is a member of the `Instrument` object.

**Type**
Read Only Boolean

**Example**
This example gets info on whether the current method has been modified or not.

\[
b\text{MethodModified} = \text{Instrument.MethodModified}
\]
MethodNew Method

Opens a new method file. This is a member of the Instrument object.

Returns
Void

Syntax
MethodNew()

Example
This example opens a new method file, thereby making it the current method:

Instrument.MethodNew()

MethodSave Method

Saves the current method file. This is a member of the Instrument object.

Returns
Boolean

Syntax
MethodSave()

Example
This example saves the current method file:

bMethodWasSaved = Instrument.MethodSave()

MethodSaveAs Method

Saves the specified method file. This is a member of the Instrument object.

Returns
Boolean

Syntax
MethodSaveAs(MethodFileName)

   MethodFileName   String. This is the name of the method file to be saved.

Example
This example saves a method file:
bMethodWasSaved = Instrument.MethodSaveAs("C:\EZChrom Elite\Method001.met")

**MethodTraces Property**

Returns the **MethodTraces** object. This is a member of the **Instrument** object.

**Type**
Read Only Object

**Example**
This example displays the name of each trace to the user:

```vba
For Each MethodTrace In Instrument.MethodTraces
    MsgBox ("Method Trace Name: " & MethodTrace.TraceName)
Next
```

**Name Property**

Returns the configured name of this instrument. This is a member of the **Instrument** object.

**Type**
Read Only String

**Example**
This example displays a message box with the configured name of this instrument:

```vba
strMsg = "The name of this Instrument is " & Instrument.Name
MsgBox (strMsg)
```

**PretreatLoad Method**

Opens the specified pretreatment file. This is a member of the **Instrument** object.

**Returns**
Boolean

**Syntax**
PretreatLoad(PretreatmentFileName)

**PretreatmentFileName**
String. This is the name of the pretreatment file to be opened.

**Example**
This example opens a pretreatment file, thereby making it the current pretreatment:

```vba
bPretreatWasOpened = Instrument.PretreatLoad("C:\EZChrom Elite\Pretreatment001.pret")
```
"C:\EZChrom Elite\Pretreat001.ape")

**PretreatmentModified Property**

Returns whether the current pretreatment has been modified or not. This is a member of the **Instrument** object.

**Type**
Read Only Boolean

**Example**
This example gets info on whether the current pretreatment has been modified or not:

```plaintext
bPretreatmentModified = Instrument.PretreatmentModified
```

**PretreatNew Method**

Opens a new pretreatment file. This is a member of the **Instrument** object.

**Returns**
Void

**Syntax**
PretreatNew()

**Example**
This example opens a new pretreatment file, thereby making it the current pretreatment:

```plaintext
Instrument.PretreatNew()
```

**PretreatSave Method**

Saves the current pretreatment file. This is a member of the **Instrument** object.

**Returns**
Boolean

**Syntax**
PretreatSave()

**Example**
This example saves the current pretreatment file:

```plaintext
bPretreatWasSaved = Instrument.PretreatSave()
```

**PretreatSaveAs Method**

Saves the specified pretreatment file. This is a member of the **Instrument** object.
Returns
Boolean

Syntax
PretreatSaveAs(PretreatmentFileName)

PretreatmentFileName
String. This is the name of the pretreatment file to be saved.

Example
This example saves a pretreatment file:

```csharp
bPretreatWasSaved = Instrument.PretreatSaveAs( _
    "C:\EZChrom Elite\Pretreat001.ape")
```

PrintCustomReport Method
Prints the current method custom report in the loaded method. This is a member of the Instrument object.

Returns
Void

Syntax
PrintCustomReport()

Example
This example loads a method and then prints its custom report:

```csharp
Instrument.CurrentMethodFile = "C:\EZChrom Elite\Method001.met"
Instrument.DataAnalysis.SubmitAnalysis
Instrument.PrintCustomReport
```

RefreshDevices Method
Refreshes the Devices object, which reloads the device list. This is a member of the Instrument object.

Returns
Void

Syntax
RefreshDevices()

Example
This example displays the most recent list of Devices:
Instrument.RefreshDevices
For Each Device In Instrument.Devices
    MsgBox Device.Name
Next

RefreshFractCollTraces Method

Refreshes the FractionCollTraces object, which reloads the FractionCollTrace list. This is a member of the Instrument object.

Returns
Void

Syntax
RefreshFractCollTraces()

Example
This example displays the most recent list of FractionCollTraces:

Instrument.RefreshFractCollTraces
For Each FractionCollTrace In Instrument.FractionCollTraces
    MsgBox FractionCollTrace.FractionID
Next

RefreshMethodChannels Method

Refreshes the MethodChannels object, which reloads the MethodChannel list. This is a member of the Instrument object.

Returns
Void

Syntax
RefreshMethodChannels()

Example
This example displays the most recent list of MethodChannels column descriptions:

Instrument.RefreshMethodChannels
For Each MethodChannel In Instrument.MethodChannels
    MsgBox MethodChannel.ColumnDescription
Next
**RefreshMethodFileInfo Method**

Refreshes the MethodFileInfo object, which reloads the MethodFileInfo object. This is a member of the Instrument object.

**Returns**
Void

**Syntax**
RefreshMethodFileInfo()

**Example**
This example shows how to refresh the MethodFileInfo:

Instrument.RefreshMethodFileInfo

**RefreshMethodTraces Method**

Refreshes the MethodTraces object, which reloads the Trace list. This is a member of the Instrument object.

**Returns**
Void

**Syntax**
RefreshMethodTraces()

**Example**
This example displays the most recent list of MethodTraces:

```
Instrument.RefreshMethodTraces
For Each MethodTrace In Instrument.MethodTraces
    MsgBox MethodTrace.TraceName
Next
```

**RefreshRunQueue Method**

Refreshes the RunQueue object, which reloads the RunQueueItem list. This is a member of the Instrument object.

**Returns**
Void

**Syntax**
RefreshRunQueue()
This example displays the most recent status of each run queue item in the collection using Next statements:

```vba
Instrument.RefreshRunQueue
nCount = Instrument.RunQueue.Count
For nIndex = 0 To nCount - 1
    MsgBox ("Run Queue Item Status: " & _
             Instrument.RunQueue(nIndex).ItemStatus)
Next nIndex
```

**RefreshTraces Method**

Refreshes the `Traces` object, which reloads the Trace list. This is a member of the `Instrument` object.

**Returns**

Void

**Syntax**

`RefreshTraces()`

**Example**

This example displays the most recent list of Traces:

```vba
Instrument.RefreshTraces
For each Trace in Instrument.Traces
    MsgBox Trace.Name
Next
```

**RunQueue Property**

Returns the `RunQueue` object. This is a member of the `Instrument` object.

**Type**

Read Only Object

**Example**

This example displays the status of each item in the run queue to the user:

```vba
Instrument.RunQueue.RefreshItemList
For Each RunQueueItem In Instrument.RunQueue
    MsgBox ("Run Queue Item Status: " & _
             RunQueueItem.ItemStatus)
Next
```

**SequenceLoad Method**

Open an existing sequence file with the specified filename. This is a member of the `Instrument` object.
Returns
Boolean

Syntax
SequenceLoad(strFileName)

Example
This example shows how to open the specified sequence file:

Instrument.SequenceLoad("C:\EZChrom Elite\Sequence001.seq")

**SequenceRun Property**

Returns the SequenceRun object. This is a member of the Instrument object.

**Type**
Read Only Object

**Example**
This example submits a sequence run:

Instrument.SequenceRun.ProbSequenceName = _
"C:\EZChrom Elite\Sequence001.seq"
Instrument.SequenceRun.SubmitSequenceRun

**SequenceSave Method**

Saves the currently loaded sequence. This is a member of the Instrument object.

**Returns**
Boolean

**Syntax**
SequenceSave()

**Example**
This example shows how to save the sequence file:

Instrument.SequenceSave

**SequenceSaveAs Method**

Saves the currently loaded sequence to a sequence file with a specified filename. This is a member of the Instrument object.

**Returns**
Boolean

**Syntax**
SequenceSaveAs(strFileName)
Example
This example shows how to save the sequence file with a specified filename:

```csharp
Instrument.SequenceSaveAs("C:\EZChrom Elite\Sequence001.seq")
```

**ShutdownOnDisconnect Property**

Sets or returns the shutdown property of this instrument. This property is only valid after a connection is made to the current instrument. By default, remote instruments on an EZServer will be left running when the client application disconnects and local instruments will be shutdown. This behavior can be changed by using this property. This is a member of the `Instrument` object.

**Type**
Read/Write Boolean

Note: If the ShutdownOnDisconnect option is not specified, it is FALSE by default and the client license associated with the instrument remains in use after the instrument is closed. Care should be used to ensure that licenses are properly released when the instrument is closed.

**Example**
This example makes sure that the current instrument is left running when the client application disconnects from it:

```csharp
Instrument.Connect
Instrument.ShutdownOnDisconnect = False
Instrument.Disconnect
```

**SingleRun Property**

Returns the `SingleRun` object. This is a member of the `Instrument` object.

**Type**
Read Only Object

**Example**
This example submits a single run:

```csharp
Instrument.SingleRun.PropMethodName = "C:\EZChrom Elite\Method001.met"
Instrument.SingleRun.PropDataName = "C:\EZChrom Elite\Data001.dat"
```

**Status Property**

Returns the status of this instrument. This is a member of the `Instrument` object.

**Type**
Read Only String

**Example**
This example displays a message box with the status of this instrument:

```
strMsg = "The status of this Instrument is " & Instrument.Status
MsgBox (strMsg)
```

**Traces Property**
Returns the [Traces](#) object. This is a member of the [Instrument](#) object.

**Type**
Read Only Object

**Example**
This example displays the name of each trace to the user:

```
For Each Trace In Instrument.Traces
    MsgBox ("Trace Name: " & Trace.Name)
Next
```

**UnlockWindow Method**
Unlocks the instrument window. This method is only valid for stand-alone systems. In a client/server environment, this method is not valid. Returns TRUE if the attempt to unlock the window was successful. This is a member of the [Instrument](#) object.

**Returns**
Boolean

**Syntax**
UnlockWindow (UserName, Password, Domain)

- **UserName** Required String. This is the user account name used to unlock the instrument window. A user account name string is not required when unlocking the instrument window if instrument login is disabled.
- **Password** Required String. This is the password for the user account used to unlock the instrument window. A password string is not required when unlocking the instrument window if instrument login is disabled.
- **Domain** Required String. This is domain of the user account used to unlock the instrument window. A domain string is not required when using the data system user account list or if instrument login is disabled.

**Example**
This example unlocks the instrument window.

```
Call Instrument.UnlockWindow(“account”, “password”, “domain”)
```
**Visible Property**

Sets or returns the visibility state of this instrument. This property is only valid for stand-alone systems. In a client/server environment, this property is not valid. This is a member of the Instrument object.

**Type**
Read/Write Boolean

**Example**
This example gets the visibility state of the current instrument:

```vbnet
bVisible = Instrument.Visible
if (bVisible = True) Then
    MsgBox ("This instrument is visible")
End If
```

This example makes the current instrument visible:

```vbnet
Instrument.Visible = True
```
SingleRun Object

This object contains the methods and properties needed to submit a single run to an instrument. The Instrument object returns a pointer to this object. For an overview of all objects, see the Object Hierarchy.

Name Property

Returns the name of this object. This is a member of the SingleRun object.

Type
Read Only String

Example
This example displays a message box with the name of this object to the user:

```
strMsg = "The name of the SingleRun object is " & _
        SingleRun.Name
MsgBox (strMsg)
```

PropAverageCalibrationReplicates Property

Sets or returns the flag that indicates if the calibration replicates should be averaged by the calibration analysis of the single run. This is a member of the SingleRun object.

Type
Read/Write Boolean

Example
This example submits a single run using the given method and data file:

```
SingleRun.PropMethodName = _
        "C:\EZChrom Elite\Method001.met"
SingleRun.PropDataName = _
        "C:\EZChrom Elite\Data001.dat"
SingleRun.PropCalibrationLevel = 2
SingleRun.PropAverageCalibrationReplicates = True
SingleRun.SubmitSingleRun
```
**PropCalibrationLevel Property**

Sets or returns the calibration level that will be used for a calibration run with the SubmitSingleRun() method. This is a member of the `SingleRun` object. A value of 0 means that no calibration will be performed.

**Type**
Read/Write Long (4 byte integer value)

**Example**
This example submits a single run using the given method and data file:

```vbscript
SingleRun.PropMethodName = _
    "C:\EZChrom Elite\Method001.met"
SingleRun.PropDataName = _
    "C:\EZChrom Elite\Data001.dat"
SingleRun.PropCalibrationLevel = 1
SingleRun.SubmitSingleRun
```

**PropClearAllCalibration Property**

Sets or returns the flag that indicates if all calibration levels should be cleared before performing the single run. This is a member of the `SingleRun` object.

**Type**
Read/Write Boolean

**Example**
This example submits a single run using the given method and data file:

```vbscript
SingleRun.PropMethodName = _
    "C:\EZChrom Elite\Method001.met"
SingleRun.PropDataName = _
    "C:\EZChrom Elite\Data001.dat"
SingleRun.PropClearAllCalibration = False
SingleRun.SubmitSingleRun
```

**PropClearCalibrationLevel Property**

Sets or returns the flag that indicates if the calibration level specified by `PropCalibrationLevel` should be cleared before performing the single run. This is a member of the `SingleRun` object.

**Type**
Read/Write Boolean

**Example**
This example submits a single run using the given method and data file:

```
SingleRun.PropMethodName = "C:\EZChrom Elite\Method001.met"
SingleRun.PropDataName = "C:\EZChrom Elite\Data001.dat"
SingleRun.PropCalibrationLevel = 2
SingleRun.PropClearCalibrationLevel = True
SingleRun.SubmitSingleRun
```

**PropClearCalibrationReplicates Property**

Sets or returns the flag that indicates if the calibration replicates should be cleared before performing the single run and calibration analysis. This is a member of the `SingleRun` object.

**Type**

Read/Write Boolean

**Example**

This example submits a single run using the given method and data file:

```
SingleRun.PropMethodName = "C:\EZChrom Elite\Method001.met"
SingleRun.PropDataName = "C:\EZChrom Elite\Data001.dat"
SingleRun.PropCalibrationLevel = 2
SingleRun.PropClearCalibrationReplicates = True
SingleRun.SubmitSingleRun
```

**PropDataName Property**

Sets or returns the data name that will be used for a single run by the `SubmitSingleRun()` method. This is a member of the `SingleRun` object.

When submitting a run using more than one repetition, it is important to be sure duplicate filenames are not used. One way to do this is to use the “incremental” filename form. “Incremental” filenames include the string “<xyz>”, where xyz is the starting three-digit repetition number. For example, a filename of “data<100>.dat” with 4 repetitions would result in four data files named “data100.dat”, “data101.dat”, “data102.dat”, and “data103.dat”. Another way would be to include the Date and Time token, <D>, in the file name.

When setting the data file name, the same tokens used by EZChrom Elite’s Single Run Dialog may be used. In addition to the Increment Number, <xyz>, and Data and Time, <D>, tokens described above, these are:

- <U> User Name
- <M> Method Name
- <I> Instrument Name
- <ID> Sample ID
Type
Read/Write String

Examples
This example submits a single run using the given method and data file:

```vba
SingleRun.PropMethodName = _
"C:\EZChrom Elite\Method001.met"
SingleRun.PropDataName = _
"C:\EZChrom Elite\Data001.dat"
SingleRun.SubmitSingleRun
```

This example submits a single run using multiple repetitions:

```vba
SingleRun.PropMethodName = _
"C:\EZChrom Elite\Method001.met"
SingleRun.PropNumberOfReps = 5
SingleRun.PropDataName = _
"C:\EZChrom Elite\Data<001>.dat"
SingleRun.SubmitSingleRun
```

**PropDilutFactor1 Property**
Sets or returns the first dilution factor that will be used for a single run by the SubmitSingleRun() method. This is a member of the `SingleRun` object. This value should be set to 1 if it is not used.

Type
Read/Write Double (8 byte floating point value)

Example
This example submits a single run using the given method and data file:

```vba
SingleRun.PropMethodName = _
"C:\EZChrom Elite\Method001.met"
SingleRun.PropDataName = _
"C:\EZChrom Elite\Data001.dat"
SingleRun.PropDilutFactor1 = 2.5
SingleRun.SubmitSingleRun
```

**PropDilutFactor2 Property**
Sets or returns the second dilution factor that will be used for a single run by the SubmitSingleRun() method. This is a member of the `SingleRun` object. This value should be set to 1 if it is not used.

Type
Read/Write Double (8 byte floating point value)
**Example**
This example submits a single run using the given method and data file:

```plaintext
SingleRun.PropMethodName = _ "C:\EZChrom Elite\Method001.met"
SingleRun.PropDataName = _ "C:\EZChrom Elite\Data001.dat"
SingleRun.PropDilutFactor2 = 2.5
SingleRun.SubmitSingleRun
```

**PropDilutFactor3 Property**
Sets or returns the third dilution factor that will be used for a single run by the SubmitSingleRun() method. This is a member of the SingleRun object. This value should be set to 1 if it is not used.

**Type**
Read/Write Double (8 byte floating point value)

**Example**
This example submits a single run using the given method and data file:

```plaintext
SingleRun.PropMethodName = _ "C:\EZChrom Elite\Method001.met"
SingleRun.PropDataName = _ "C:\EZChrom Elite\Data001.dat"
SingleRun.PropDilutFactor3 = 2.5
SingleRun.SubmitSingleRun
```

**PropInitialVial Property**
Sets or returns the vial that will be used for the first injection of a sample in a single run by the SubmitSingleRun() method. This is a member of the SingleRun object.

To specify no vial, the `ezAutosamplerNoVial` constant may be used.

Vials are specified using the encoded numeric vial identifier used internally by Elite. This numeric vial identifier corresponds to the formatted, decoded string of the autosampler user interface. In order to work with vial identifiers using a formatted, decoded string, use `PropInitialVialText` instead.

**Type**
Read/Write Long (4 byte integer value)

**Example**
This example submits a single run using the given method and data file:

```plaintext
SingleRun.PropMethodName = _
```
"C:\EZChrom Elite\Method001.met"
SingleRun.PropDataName = _
    "C:\EZChrom Elite\Data001.dat"
SingleRun.PropInitialVial = 5
SingleRun.SubmitSingleRun

**PropInitialVialText Property**

Sets or returns the vial that will be used for the first injection of a sample in a single run by the SubmitSingleRun() method. This is a member of the SingleRun object.

To specify no vial, an empty string may be used.

Vials are specified using the formatted, decoded string of the autosampler user interface. This vial string corresponds to the encoded numeric vial identifier used internally by Elite. In order to work with vial identifiers using a numeric format, use PropInitialVial instead.

**Type**
Read/Write String

**Example**
This example submits a single run using the given method and data file:

    SingleRun.PropMethodName = _
        "C:\EZChrom Elite\Method001.met"
    SingleRun.PropDataName = _
        "C:\EZChrom Elite\Data001.dat"
    SingleRun.PropInitialVialText = "A01"
    SingleRun.SubmitSingleRun

**PropInjectionVolume Property**

Sets or returns the injection volume that will be used for a run with the SubmitSingleRun() method. This is a member of the SingleRun object. This value can be set to 0 if it is not used.

**Type**
Read/Write Single (4 byte floating point value)

**Example**
This example submits a single run using the given method and data file:

    SingleRun.PropMethodName = _
        "C:\EZChrom Elite\Method001.met"
    SingleRun.PropDataName = _
        "C:\EZChrom Elite\Data001.dat"
    SingleRun.PropInjectionVolume = 10.5
    SingleRun.SubmitSingleRun
**PropISTDAmount Property**

Sets or returns the ISTD amount that will be used for a single run by the SubmitSingleRun() method. This is a member of the `SingleRun` object. The ISTD amount is a number that specifies the amount of Internal Standard in your unknown sample. This value should be set to 1 if it is not used.

**Type**
Read/Write Single (4 byte floating point value)

**Example**
This example submits a single run using the given method and data file:

```plaintext
SingleRun.PropMethodName = "C:\EZChrom Elite\Method001.met"
SingleRun.PropDataName = "C:\EZChrom Elite\Data001.dat"
SingleRun.PropISTDAmount = 10.5
SingleRun.SubmitSingleRun
```

**PropMethodName Property**

Sets or returns the method name that will be used for a single run by the SubmitSingleRun() method. This is a member of the `SingleRun` object.

**Type**
Read/Write String

**Example**
This example submits a single run using the given method and data file:

```plaintext
SingleRun.PropMethodName = "C:\EZChrom Elite\Method001.met"
SingleRun.PropDataName = "C:\EZChrom Elite\Data001.dat"
SingleRun.SubmitSingleRun
```

**PropMultFactor1 Property**

Sets or returns the first multiplication factor that will be used for a single run by the SubmitSingleRun() method. This is a member of the `SingleRun` object. All quantitated peaks will be multiplied by this factor. This value should be set to 1 if it is not used.

**Type**
Read/Write Double (8 byte floating point value)

**Example**
This example submits a single run using the given method and data file:

```plaintext
SingleRun.PropMethodName = "C:\EZChrom Elite\Method001.met"
SingleRun.PropDataName = "C:\EZChrom Elite\Data001.dat"
SingleRun.PropMultFactor1 = 2.5
SingleRun.SubmitSingleRun
```

### PropMultFactor2 Property

Sets or returns the second multiplication factor that will be used for a single run by the SubmitSingleRun() method. This is a member of the SingleRun object. All quantitated peaks will be multiplied by this factor. This value should be set to 1 if it is not used.

**Type**

Read/Write Double (8 byte floating point value)

**Example**

This example submits a single run using the given method and data file:

```plaintext
SingleRun.PropMethodName = "C:\EZChrom Elite\Method001.met"
SingleRun.PropDataName = "C:\EZChrom Elite\Data001.dat"
SingleRun.PropMultFactor2 = 2.5
SingleRun.SubmitSingleRun
```

### PropMultFactor3 Property

Sets or returns the third multiplication factor that will be used for a single run by the SubmitSingleRun() method. This is a member of the SingleRun object. All quantitated peaks will be multiplied by this factor. This value should be set to 1 if it is not used.

**Type**

Read/Write Double (8 byte floating point value)

**Example**

This example submits a single run using the given method and data file:

```plaintext
SingleRun.PropMethodName = "C:\EZChrom Elite\Method001.met"
SingleRun.PropDataName = "C:\EZChrom Elite\Data001.dat"
SingleRun.PropMultFactor3 = 2.5
SingleRun.SubmitSingleRun
```
**PropMultiplicationFactor Property**

This property is now obsolete. Please use the new PropMultFactor# properties. This is a member of the SingleRun object.

**Type**
Read/Write Single (4 byte floating point value)

**PropPrintCalibrationReport Property**

Sets or returns the flag that indicates if a calibration report with response factors should be printed after performing the single run and calibration analysis. This is a member of the SingleRun object.

**Type**
Read/Write Boolean

**Example**
This example submits a single run using the given method and data file:

```vba
SingleRun.PropMethodName = "C:\EZChrom Elite\Method001.met"
SingleRun.PropDataName = "C:\EZChrom Elite\Data001.dat"
SingleRun.PropCalibrationLevel = 2
SingleRun.PropPrintCalibrationReport = True
SingleRun.SubmitSingleRun
```

**PropNumberOfReps Property**

Sets or returns the number of repetitions for this run. This is a member of the SingleRun object.

When using more than one repetition, it is important to remember to use the "incremental" filename format for the datafile. See the PropDataName property of the SingleRun object for more information.

**Type**
Read/Write Integer (2-byte Integer value)

**Example**
This example gets the number of repetitions for the run:

```vba
Dim nReps as Integer
nReps = SingleRun.PropNumberOfReps
```

This example sets the number of repetitions for the run:

```vba
SingleRun.PropNumberOfReps = 5
SingleRun.PropDataName = "C:\EZChrom Elite\Data<001>.dat"
```
PropProject Property

Gets or sets the project that will be used when performing the run. It is only necessary to specifically set the project when submitting a run from outside of the current project. This is a member of the SingleRun object.

Type
Read/Write String

Example
This example gets the project for the run:

```vba
Dim strProject as String
strProject = SingleRun.PropProject
```

This example sets the project for the run:

```vba
SingleRun.PropProject = strProject
```

PropPrintCustomMethodReport Property

Sets or returns the flag that indicates if the custom method report should be printed after performing the single run. This is a member of the SingleRun object.

Type
Read/Write Boolean

Example
This example submits a single run using the given method and data file:

```vba
SingleRun.PropMethodName = _
    "C:\EZChrom Elite\Method001.met"
SingleRun.PropDataName = _
    "C:\EZChrom Elite\Data001.dat"
SingleRun.PropPrintCustomMethodReport = True
SingleRun.SubmitSingleRun
```

PropSampleAmount Property

Sets or returns the sample amount that will be used for a single run by the SubmitSingleRun() method. This is a member of the SingleRun object. The sample amount is a number that is used for calculation purposes. The sample amount value is used as a divisor during calculation of concentrations. It is intended to compensate for differences between samples due to weighting and when percentages of the total sample are being calculated rather than the amount detected in an injection. This value should be set to 1 if it is not used.

Type
Read/Write Single (4 byte floating point value)

Example
This example submits a single run using the given method and data file:
SingleRun.PropMethodName = "C:\EZChrom Elite\Method01.met"
SingleRun.PropDataName = "C:\EZChrom Elite\Data001.dat"
SingleRun.PropSampleAmount = 10.5
SingleRun.SubmitSingleRun

**PropSampleDescription Property**

Sets or returns the sample description that will be used for a single run by the SubmitSingleRun() method. This is a member of the SingleRun object.

**Type**
Read/Write String

**Example**
This example submits a single run using the given method and data file:

```java
SingleRun.PropMethodName = "C:\EZChrom Elite\Method01.met"
SingleRun.PropDataName = "C:\EZChrom Elite\Data001.dat"
SingleRun.PropSampleDescription = "This is a test sample."
SingleRun.SubmitSingleRun
```

**PropSampleID Property**

Sets or returns the sample id that will be used for a single run by the SubmitSingleRun() method. This is a member of the SingleRun object.

**Type**
Read/Write String

When setting the sample ID, the same tokens used by EZChrom Elite’s Single Run Dialog may be used. These are:

- `<001>` Increment Number
- `<U>` User Name
- `<M>` Method Name
- `<I>` Instrument Name
- `<D>` Date and Time

**Example**
This example submits a single run using the given method and data file:

```java
SingleRun.PropMethodName = "C:\EZChrom Elite\Method01.met"
SingleRun.PropDataName = "C:\EZChrom Elite\Data001.dat"
SingleRun.PropSampleID = "Test Sample"
```
PropTowerToProcess Property

Sets or returns the tower number that will be used for the single run. The tower number is a 1-based number representing the desired tower. For example, in a dual tower instrument the front tower is tower number 1 and the rear tower is tower number 2. This is a member of the SingleRun object.

Type
Read/Write Long (4 byte integer value)

Example
This example submits a single run on the front tower using the given method and data file:
```
SingleRun.PropMethodName = 
    "C:\EZChrom Elite\Method001.met"
SingleRun.PropDataName = 
    "C:\EZChrom Elite\Data001.dat"
SingleRun.PropTowerToProcess = 1
SingleRun.SubmitSingleRun
```

PropUserDomain Property

Sets or returns the domain of the user that will be used to submit the single run. If this property is not set before a run is made, then the domain of the user that connected to this instrument will be used for the run. This is a member of the SingleRun object.

Type
Read/Write String

Example
This example submits a single run using the given method and data file:
```
SingleRun.PropMethodName = 
    "C:\EZChrom Elite\Method001.met"
SingleRun.PropDataName = 
    "C:\EZChrom Elite\Data001.dat"
SingleRun.PropUserName = "User A"
SingleRun.PropUserPassword = "xyz"
SingleRun.PropUserDomain = "TEST"
SingleRun.SubmitSingleRun
```

PropUserName Property

Sets or returns the name of the user that will be used to submit the single run. If this property is not set before a run is made, then the user that connected to this instrument will be used for the run. This is a member of the SingleRun object.
Read/Write String

Example
This example submits a single run using the given method and data file:

```python
SingleRun.PropMethodName = "C:\EZChrom Elite\Method001.met"
SingleRun.PropDataName = "C:\EZChrom Elite\Data001.dat"
SingleRun.PropUserName = "User A"
SingleRun.PropUserPassword = "xyz"
SingleRun.PropUserDomain = "TEST"
SingleRun.SubmitSingleRun
```

**PropUserPassword Property**

Sets or returns the password for the user that will be used to submit the single run. If this property is not set before a run is made, then the password of the user that connected to this instrument will be used for the run. This is a member of the `SingleRun` object.

**Type**

Read/Write String

Example
This example submits a single run using the given method and data file:

```python
SingleRun.PropMethodName = "C:\EZChrom Elite\Method001.met"
SingleRun.PropDataName = "C:\EZChrom Elite\Data001.dat"
SingleRun.PropUserName = "User A"
SingleRun.PropUserPassword = "xyz"
SingleRun.PropUserDomain = "TEST"
SingleRun.SubmitSingleRun
```

**SetDefaultRunProperties Method**

Sets all of the properties for a single run back to their default settings. This is a member of the `SingleRun` object.

The default settings for the single run properties are as follows:

<table>
<thead>
<tr>
<th>Run Property</th>
<th>Default Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropAverageCalibrationReplicates</td>
<td>False</td>
</tr>
<tr>
<td>PropCalibrationLevel</td>
<td>0</td>
</tr>
<tr>
<td>PropClearAllCalibration</td>
<td>False</td>
</tr>
<tr>
<td>PropClearCalibrationLevel</td>
<td>False</td>
</tr>
<tr>
<td>PropClearCalibrationReplicates</td>
<td>False</td>
</tr>
</tbody>
</table>
PropDataName ""
PropInitialVial ezAutosamplerNoVial
PropInjectionVolume 0.0
PropISTDAmount 1.0
PropMethodName ""
PropMultiplicationFactor 1.0
PropPrintCalibrationReport False
PropPrintCustomMethodReport False
PropSampleAmount 1.0
PropSampleDescription ""
PropSampleID ""
PropTowerToProcess 1
PropUserDomain Domain of user connected to instrument
PropUserName Name of user connected to instrument
PropUserPassword Password of user connected to instrument

Returns
Void

Syntax
SetDefaultRunProperties()

Example
This example submits a single run using the given method and data file:

```python
SingleRun.SetDefaultRunProperties
SingleRun.PropMethodName = "C:\EZChrom Elite\Method001.met"
SingleRun.PropDataName = "C:\EZChrom Elite\Data001.dat"
SingleRun.SubmitSingleRun
```

SubmitSingleRun Method

Submits a single run to the run queue of the connected instrument. All of the run properties are used to control how the run is made. This is a member of the SingleRun object.

Note: To prevent Elite from appending ".dat" to the resulting data file name, append a single "." to the data file name given as PropDataName. For example, providing "data" will result in data.dat as the file name and providing "data." will result in data as the file name.

Returns
Void

Syntax
SubmitSingleRun()

Example
This example submits a single run using the given method and data file:
SubmitPriorityRun Method

Submits a priority run to the run queue of the connected instrument. All of the run properties are used to control how the run is made. This is a member of the SingleRun object.

Returns
Void

Syntax
SubmitPriorityRun()

Example
This example submits a priority run using the given method and data file:

```csharp
SingleRun.SetDefaultRunProperties
SingleRun.PropMethodName = _
    "C:\EZChrom Elite\Method001.met"
SingleRun.PropDataName = _
    "C:\EZChrom Elite\Data001.dat"
SingleRun.SubmitPriorityRun
```
**SequenceRun Object**

This object contains the methods and properties needed to submit a sequence run to an instrument. The Instrument object returns a pointer to this object. For an overview of all objects, see the Object Hierarchy.

**Analyze Method**

Performs reprocessing of previously acquired sequence data. All of the sequence run properties are used to control how the analysis is performed. This is a member of the SequenceRun object.

**Returns**

Void

**Syntax**

Analyze()

**Example**

This example reprocesses a sequence run using the given sequence file:

```vba
SequenceRun.SetDefaultSequenceProperties
SequenceRun.PropSequenceName = _
   "C:\EZChrom Elite\Sequence001.seq"
SequenceRun.Analyze
```

**Name Property**

Returns the name of this object. This is a member of the SequenceRun object.

**Type**

Read Only String

**Example**

This example displays a message box with the name of this object to the user:

```vba
strMsg = "The name of the SequenceRun object is " & _
   SequenceRun.Name
MsgBox (strMsg)
```

**PropBracketingMode Property**

Sets or returns a flag that specifies the sequence bracketing mode that will be used for a sequence run by the SubmitSequenceRun() method. This is a member of the SequenceRun object.

The bracketing mode is one of the following EZSequenceBracketingFlags constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ezSequenceBracketingNone</td>
<td>No sequence bracketing will be used.</td>
</tr>
<tr>
<td>ezSequenceBracketingStandard</td>
<td>Use the standard mode of bracketing the sequence.</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>ezSequenceBracketingStandardClearCalib</td>
<td>Use the standard mode of bracketing the sequence and clear the calibration before the start of each calibration set.</td>
</tr>
<tr>
<td>ezSequenceBracketingSequence</td>
<td>Use the sequence mode of bracketing the sequence.</td>
</tr>
<tr>
<td>ezSequenceBracketingSequenceBackCalculate</td>
<td>Use the sequence mode of bracketing the sequence and back calculate the calibration runs.</td>
</tr>
</tbody>
</table>

**Type**
Read/Write Long (4 byte integer value)

**Example**
This example submits a sequence run using the given sequence file:

```vbscript
SequenceRun.PropSequenceName = _
    "C:\EZChrom Elite\Sequence001.seq"
SequenceRun.PropBracketingMode = ezSequenceBracketingNone
SequenceRun.SubmitSequenceRun
```

### PropEmailOnStart Property

**Gets or sets the state of the “e-mail on start of run” flag.** This is a member of the `SequenceRun` object.

**Type**
Read/Write Boolean

**Example**
This example gets the state of the “e-mail on start of run” flag:

```vbscript
Dim bEnabled as Boolean
bEnabled = SequenceRun.PropEmailOnStart
```

This example sets the state of the “e-mail on start of run” flag:

```vbscript
SequenceRun.PropEmailOnStart = TRUE
```

### PropEmailOnStopOrError Property

**Gets or sets the state of the “e-mail on run stop or run error” flag.** This is a member of the `SequenceRun` object.

**Type**
Read/Write Boolean

**Example**
This example gets the state of the “e-mail on run stop or run error” flag:

```vba
Dim bEnabled as Boolean
bEnabled = SequenceRun.PropEmailOnStopOrError
```

This example sets the state of the “e-mail on run stop or run error” flag:

```vba
SequenceRun.PropEmailOnStopOrError = TRUE
```

**PropEmailToAddress Property**

Gets or sets the state of the e-mail recipient list. This is a member of the `SequenceRun` object.

**Type**
Read/Write String

**Example**
This example gets the state of the “email on run stop or run error” flag:

```vba
Dim strRecipients as String
strRecipients = SequenceRun.PropEmailToAddress
```

This example sets the state of the “email on run stop or run error” flag:

```vba
SequenceRun.PropEmailToAddress = strRecipients
```

**PropPrintCalibrationReport Property**

Sets or returns the flag that indicates if the sequence reports should be printed while processing the sequence. This is a member of the `SequenceRun` object.

**Type**
Read/Write Boolean

**Example**
This example submits a sequence run using the given sequence file. As the sequence is processed, the sequence reports will be printed:

```vba
SequenceRun.PropSequenceName = "C:\EZChrom Elite\Sequence001.seq"
SequenceRun.PropPrintCalibrationReport = True
SequenceRun.SubmitSequenceRun
```
PropPrintCustomMethodReport Property

Sets or returns the flag that indicates if the custom method report should be printed after performing the sequence run. This is a member of the SequenceRun object.

Type
Read/Write Boolean

Example
This example submits a sequence run using the given sequence file:

```vbnet
SequenceRun.PropSequenceName = "C:\EZChrom Elite\Sequence001.seq"
SequenceRun.PropPrintCustomMethodReport = True
SequenceRun.SubmitSequenceRun
```

PropProcessingRunMode Property

Sets or returns a flag that specifies the sequence processing run mode that will be used for a sequence run by the SubmitSequenceRun() method. This is a member of the SequenceRun object.

The processing run mode is one of the following EZSequenceRunFlags constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ezSequenceRunNormal</td>
<td>Perform a normal sequence run.</td>
</tr>
</tbody>
</table>

Type
Read/Write Long (4 byte integer value)

Example
This example submits a sequence run using the given sequence file:

```vbnet
SequenceRun.PropSequenceName = "C:\EZChrom Elite\Sequence001.seq"
SequenceRun.PropProcessingRunMode = ezSequenceRunNormal
SequenceRun.SubmitSequenceRun
```

PropProject Property

Gets or sets the project that will be used when running the sequence. It is only necessary to specifically set the project when submitting a run from outside of the current project. This is a member of the SequenceRun object.

Type
Read/Write String

Example
This example gets the project for the sequence run:
```
Dim strProject as String
strProject = SequenceRun.PropProject

This example sets the project for the sequence run:

    SequenceRun.PropProject = strProject
```

**PropRangeMode Property**

Sets or returns a flag that specifies the range mode that will be used for a sequence run by the SubmitSequenceRun() method. This is a member of the `SequenceRun` object.

The range mode is one of the following `EZSequenceRangeFlags` constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ezSequenceRangeAll</td>
<td>All lines of the sequence will be processed.</td>
</tr>
<tr>
<td>ezSequenceRangeSpecified</td>
<td>Only the specified lines of the sequence will be processed. For a description of how to specify a sequence range, see the <code>PropRangeSpecifier</code> property.</td>
</tr>
</tbody>
</table>

**Type**

Read/Write Long (4 byte integer value)

**Example**

This example submits a sequence run using the given sequence file:

```
    SequenceRun.PropSequenceName = _
        "C:\EZChrom Elite\Sequence001.seq"
    SequenceRun.PropRangeMode = ezSequenceRangeAll
    SequenceRun.SubmitSequenceRun
```

**PropRangeSpecifier Property**

Sets or returns a string that specifies the range of lines that will be processed in a sequence run by the SubmitSequenceRun() method. This is a member of the `SequenceRun` object. The `PropRangeMode` property must be set to `ezSequenceRangeSpecified` before this property will be used. The syntax of this string is described in the data system online help file.

**Type**

Read/Write String

**Example**

This example submits a sequence run using the given sequence file. Lines 4, 5, and 6 of the sequence will be processed:

```
    SequenceRun.PropSequenceName = _
        "C:\EZChrom Elite\Sequence001.seq"
```
PropSequenceName Property

Sets or returns the sequence file name that will be used for a sequence run by the SubmitSequenceRun() method. This is a member of the SequenceRun object.

Type
Read/Write String

Example
This example submits a sequence run using the given sequence file:

```vba
SequenceRun.PropSequenceName = "C:\EZChrom Elite\Sequence001.seq"
SequenceRun.SubmitSequenceRun
```

PropTowerToProcess Property

Sets or returns the tower number that will be used for the sequence run. The tower number is a 1-based number representing the desired tower. To indicate all towers, use -1. For example, in a dual tower instrument the front tower is tower number 1 and the rear tower is tower number 2. This is a member of the SequenceRun object.

Type
Read/Write Long (4 byte integer value)

Example
This example submits a sequence run on both towers using the given sequence file:

```vba
SequenceRun.PropSequenceName = "C:\EZChrom Elite\Sequence001.seq"
SequenceRun.PropTowerToProcess = -1
SequenceRun.SubmitSequenceRun
```

PropUserDomain Property

Sets or returns the user’s domain that will be used to submit the sequence run. If this property is not set before a run is made, then the domain of the user that connected to this instrument will be used for the run. This is a member of the SequenceRun object.

Type
Read/Write String

Example
This example submits a sequence run using the given sequence file:

```plaintext
SequenceRun.PropSequenceName = "C:\EZChrom Elite\Sequence001.seq"
SequenceRun.PropUserName = "User A"
SequenceRun.PropUserPassword = "xyz"
SequenceRun.PropUserDomain = "TEST"
SequenceRun.SubmitSequenceRun
```

PropUserName Property
Sets or returns the name of the user that will be used to submit the sequence run. If this property is not set before a run is made, then the user that connected to this instrument will be used for the run. This is a member of the SequenceRun object.

Type
Read/Write String

Example
This example submits a sequence run using the given sequence file:

```plaintext
SequenceRun.PropSequenceName = "C:\EZChrom Elite\Sequence001.seq"
SequenceRun.PropUserName = "User A"
SequenceRun.PropUserPassword = "xyz"
SequenceRun.PropUserDomain = "TEST"
SequenceRun.SubmitSequenceRun
```

PropUserPassword Property
Sets or returns the password for the user that will be used to submit the sequence run. If this property is not set before a run is made, then the password of the user that connected to this instrument will be used for the run. This is a member of the SequenceRun object.

Type
Read/Write String

Example
This example submits a sequence run using the given sequence file:

```plaintext
SequenceRun.PropSequenceName = "C:\EZChrom Elite\Sequence001.seq"
SequenceRun.PropUserName = "User A"
SequenceRun.PropUserPassword = "xyz"
SequenceRun.PropUserDomain = "TEST"
SequenceRun.SubmitSequenceRun
```
SetDefaultSequenceProperties Method

Sets all of the properties for a sequence run back to their default settings. This is a member of the SequenceRun object.

The default settings for the sequence run properties are as follows:

<table>
<thead>
<tr>
<th>Run Property</th>
<th>Default Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropBracketingMode</td>
<td>ezSequenceBracketingNone</td>
</tr>
<tr>
<td>PropPrintCalibrationReport</td>
<td>False</td>
</tr>
<tr>
<td>PropPrintCustomMethodReport</td>
<td>False</td>
</tr>
<tr>
<td>PropProcessingRunMode</td>
<td>ezSequenceRunNormal</td>
</tr>
<tr>
<td>PropRangeMode</td>
<td>ezSequenceRangeAll</td>
</tr>
<tr>
<td>PropRangeSpecifier</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>PropSequenceName</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>PropTowerToProcess</td>
<td>1</td>
</tr>
<tr>
<td>PropUserDomain</td>
<td>Domain of user connected to instrument</td>
</tr>
<tr>
<td>PropUserName</td>
<td>Name of user connected to instrument</td>
</tr>
<tr>
<td>PropUserPassword</td>
<td>Password of user connected to instrument</td>
</tr>
</tbody>
</table>

Returns
Void

Syntax
SetDefaultSequenceProperties()

Example
This example submits a sequence run using the given sequence file:

```vbscript
SequenceRun.SetDefaultSequenceProperties
SequenceRun.PropSequenceName = "C:\EZChrom Elite\Sequence001.seq"
SequenceRun.SubmitSequenceRun
```

SubmitSequenceRun Method

Submits a sequence run to the run queue of the connected instrument. All of the sequence run properties are used to control how the run is made. This is a member of the SequenceRun object.

Returns
Void

Syntax
SubmitSequenceRun()

Example
This example submits a sequence run using the given sequence file:

```vbscript
SequenceRun.SetDefaultSequenceProperties
SequenceRun.PropSequenceName = "C:\EZChrom Elite\Sequence001.seq"
SequenceRun.SubmitSequenceRun
```
DataAnalysis Object

This object contains the methods and properties needed to perform an analysis of a specified data file. The instrument object returns a pointer to this object. For an overview of all objects, see the Object Hierarchy.

Name Property

Returns the name of this object. This is a member of the DataAnalysis object.

Type
Read Only String

Example
This example displays a message box with the name of this object to the user:

```vba
strMsg = "The name of the DataAnalysis object is " & _
         DataAnalysis.Name
MsgBox (strMsg)
```

PropAverageCalibrationReplicates Property

Sets or returns the flag that indicates if the calibration replicates should be averaged by the calibration analysis. This is a member of the DataAnalysis object.

Type
Read/Write Boolean

Example
```vba
DataAnalysis.PropMethodName = _
    "C:\EZChrom Elite\Method001.met"
DataAnalysis.PropDataName = _
    "C:\EZChrom Elite\Data001.dat"
DataAnalysis.PropCalibrationLevel = 1
DataAnalysis.PropAverageCalibrationReplicates = True
DataAnalysis.SubmitAnalysis(ezAnalysisSpecifiedOptions)
```

PropCalibrationLevel Property

Sets or returns the calibration level that will be used for an analysis. This is a member of the DataAnalysis object. A value of 0 means that no calibration will be performed.

Type
Read/Write Long (4 byte integer value)

Example
This example performs an analysis using the given method and data file:

```csharp
DataAnalysis.PropMethodName = "C:\EZChrom Elite\Method001.met"
DataAnalysis.PropDataName = "C:\EZChrom Elite\Data001.dat"
DataAnalysis.PropCalibrationLevel = 1
DataAnalysis.SubmitAnalysis(ezAnalysisSpecifiedOptions)
```

**PropClearAllCalibration Property**

Sets or returns the flag that indicates if all calibration levels should be cleared before performing the analysis. This is a member of the `DataAnalysis` object.

**Type**
Read/Write Boolean

**Example**
This example performs an analysis using the given method and data file:

```csharp
DataAnalysis.PropMethodName = "C:\EZChrom Elite\Method001.met"
DataAnalysis.PropDataName = "C:\EZChrom Elite\Data001.dat"
DataAnalysis.PropCalibrationLevel = 2
DataAnalysis.PropClearAllCalibration = False
DataAnalysis.SubmitAnalysis(ezAnalysisSpecifiedOptions)
```

**PropClearCalibrationLevel Property**

Sets or returns the flag that indicates if the calibration level specified by `PropCalibrationLevel` should be cleared before performing the analysis. This is a member of the `DataAnalysis` object.

**Type**
Read/Write Boolean

**Example**
This example performs an analysis using the given method and data file:

```csharp
DataAnalysis.PropMethodName = "C:\EZChrom Elite\Method001.met"
DataAnalysis.PropDataName = "C:\EZChrom Elite\Data001.dat"
DataAnalysis.PropCalibrationLevel = 2
DataAnalysis.PropClearCalibrationLevel = True
DataAnalysis.SubmitAnalysis(ezAnalysisSpecifiedOptions)
```
**PropClearCalibrationReplicates Property**

Sets or returns the flag that indicates if the calibration replicates should be cleared before performing the analysis. This is a member of the `DataAnalysis` object.

**Type**
Read/Write Boolean

**Example**
This example performs an analysis using the given method and data file:

```
DataAnalysis.PropMethodName = "C:\EZChrom Elite\Method001.met"
DataAnalysis.PropDataName = "C:\EZChrom Elite\Data001.dat"
DataAnalysis.PropCalibrationLevel = 2
DataAnalysis.PropClearCalibrationReplicates = True
DataAnalysis.SubmitAnalysis(ezAnalysisSpecifiedOptions)
```

**PropDataName Property**

Sets or returns the data name that will be used for an analysis. This is a member of the `DataAnalysis` object.

**Type**
Read/Write String

**Example**
This example performs an analysis using the given method and data file:

```
DataAnalysis.PropMethodName = "C:\EZChrom Elite\Method001.met"
DataAnalysis.PropDataName = "C:\EZChrom Elite\Data001.dat"
DataAnalysis.SubmitAnalysis(ezAnalysisSpecifiedOptions)
```

**PropDilutFactor1 Property**

Sets or returns the first dilution factor that will be used for an analysis. This is a member of the `DataAnalysis` object. This value should be set to 1 if it is not used.

**Type**
Read/Write Double (8 byte floating point value)

**Example**
This example performs an analysis using the given method and data file:

```
DataAnalysis.PropMethodName = _
                      "C:\EZChrom Elite\Method001.met"
DataAnalysis.PropDataName = _
                      "C:\EZChrom Elite\Data001.dat"
DataAnalysis.PropDilutFactor1 = 2.5
DataAnalysis.SubmitAnalysis(ezAnalysisSpecifiedOptions)
```

**PropDilutFactor2 Property**

Sets or returns the second dilution factor that will be used for an analysis. This is a member of the `DataAnalysis` object. This value should be set to 1 if it is not used.

**Type**

Read/Write Double (8 byte floating point value)

**Example**

This example performs an analysis using the given method and data file:

```
DataAnalysis.PropMethodName = _
                      "C:\EZChrom Elite\Method001.met"
DataAnalysis.PropDataName = _
                      "C:\EZChrom Elite\Data001.dat"
DataAnalysis.PropDilutFactor2 = 2.5
DataAnalysis.SubmitAnalysis(ezAnalysisSpecifiedOptions)
```

**PropDilutFactor3 Property**

Sets or returns the third dilution factor that will be used for an analysis. This is a member of the `DataAnalysis` object. This value should be set to 1 if it is not used.

**Type**

Read/Write Double (8 byte floating point value)

**Example**

This example performs an analysis using the given method and data file:

```
DataAnalysis.PropMethodName = _
                      "C:\EZChrom Elite\Method001.met"
DataAnalysis.PropDataName = _
                      "C:\EZChrom Elite\Data001.dat"
DataAnalysis.PropDilutFactor3 = 2.5
DataAnalysis.SubmitAnalysis(ezAnalysisSpecifiedOptions)
```
**PropISTDAmount Property**

Sets or returns the ISTD amount that will be used for an analysis. This is a member of the DataAnalysis object. The ISTD amount is a number that specifies the amount of Internal Standard in your unknown sample. This value should be set to 1 if it is not used.

**Type**
Read/Write Single (4 byte floating point value)

**Example**
This example performs an analysis using the given method and data file:

```vbnet
DataAnalysis.PropMethodName = _
    "C:\EZChrom Elite\Method001.met"
DataAnalysis.PropDataName = _
    "C:\EZChrom Elite\Data001.dat"
DataAnalysis.PropISTDAmount = 10.5
DataAnalysis.SubmitAnalysis(ezAnalysisSpecifiedOptions)
```

**PropMethodName Property**

Sets or returns the method name that will be used for an analysis. This is a member of the DataAnalysis object.

**Type**
Read/Write String

**Example**
This example performs an analysis using the given method and data file:

```vbnet
DataAnalysis.PropMethodName = _
    "C:\EZChrom Elite\Method001.met"
DataAnalysis.PropDataName = _
    "C:\EZChrom Elite\Data001.dat"
DataAnalysis.SubmitAnalysis(ezAnalysisSpecifiedOptions)
```

**PropMultFactor1 Property**

Sets or returns the first multiplication factor that will be used for an analysis. This is a member of the DataAnalysis object. All quantitated peaks will be multiplied by this factor. This value should be set to 1 if it is not used.

**Type**
Read/Write Double (8 byte floating point value)

**Example**
This example performs an analysis using the given method and data file:

```
DataAnalysis.PropMethodName = "C:\EZChrom Elite\Method001.met"
DataAnalysis.PropDataName = "C:\EZChrom Elite\Data001.dat"
DataAnalysis.PropMultFactor1 = 2.5
DataAnalysis.SubmitAnalysis(ezAnalysisSpecifiedOptions)
```

### PropMultFactor2 Property

Sets or returns the second multiplication factor that will be used for an analysis. This is a member of the DataAnalysis object. All quantitated peaks will be multiplied by this factor. This value should be set to 1 if it is not used.

**Type**

Read/Write Double (8 byte floating point value)

**Example**

This example performs an analysis using the given method and data file:

```
DataAnalysis.PropMethodName = "C:\EZChrom Elite\Method001.met"
DataAnalysis.PropDataName = "C:\EZChrom Elite\Data001.dat"
DataAnalysis.PropMultFactor2 = 2.5
DataAnalysis.SubmitAnalysis(ezAnalysisSpecifiedOptions)
```

### PropMultFactor3 Property

Sets or returns the third multiplication factor that will be used for an analysis. This is a member of the DataAnalysis object. All quantitated peaks will be multiplied by this factor. This value should be set to 1 if it is not used.

**Type**

Read/Write Double (8 byte floating point value)

**Example**

This example performs an analysis using the given method and data file:

```
DataAnalysis.PropMethodName = "C:\EZChrom Elite\Method001.met"
DataAnalysis.PropDataName = "C:\EZChrom Elite\Data001.dat"
DataAnalysis.PropMultFactor3 = 2.5
DataAnalysis.SubmitAnalysis(ezAnalysisSpecifiedOptions)
```
**PropMultiplicationFactor Property**

This property is now obsolete. Please use the new PropMultFactor# properties. This is a member of the **DataAnalysis** object.

**Type**
Read/Write Single (4 byte floating point value)

**PropPrintCalibrationReport Property**

Sets or returns the flag that indicates if a calibration report with response factors should be printed after performing the analysis. This is a member of the **DataAnalysis** object.

**Type**
Read/Write Boolean

**Example**
This example performs an analysis using the given method and data file:

```vbnet
DataAnalysis.PropMethodName = "C:\EZChrom Elite\Method001.met"
DataAnalysis.PropDataName = "C:\EZChrom Elite\Data001.dat"
DataAnalysis.PropCalibrationLevel = 2
DataAnalysis.PropPrintCalibrationReport = True
DataAnalysis.SubmitAnalysis(ezAnalysisSpecifiedOptions)
```

**PropPrintCustomMethodReport Property**

Sets or returns the flag that indicates if the custom method report should be printed after performing the analysis. This is a member of the **DataAnalysis** object.

**Type**
Read/Write Boolean

**Example**
This example performs an analysis using the given method and data file:

```vbnet
DataAnalysis.PropMethodName = "C:\EZChrom Elite\Method001.met"
DataAnalysis.PropDataName = "C:\EZChrom Elite\Data001.dat"
DataAnalysis.PropPrintCustomMethodReport = True
DataAnalysis.SubmitAnalysis(ezAnalysisSpecifiedOptions)
```
**PropSampleAmount Property**

Sets or returns the sample amount that will be used for an analysis. This is a member of the `DataAnalysis` object. The sample amount is a number that is used for calculation purposes. The sample amount value is used as a divisor during calculation of concentrations. It is intended to compensate for differences between samples due to weighting and when percentages of the total sample are being calculated rather than the amount detected in an injection. This value should be set to 1 if it is not used.

**Type**
Read/Write Single (4 byte floating point value)

**Example**
This example performs an analysis using the given method and data file:

```python
DataAnalysis.PropMethodName = _
    "C:\EZChrom Elite\Method001.met"
DataAnalysis.PropDataName = _
    "C:\EZChrom Elite\Data001.dat"
DataAnalysis.PropSampleAmount = 10.5
DataAnalysis.SubmitAnalysis(ezAnalysisSpecifiedOptions)
```

**PropSampleID Property**

Sets or returns the sample id that will be used for an analysis. This is a member of the `DataAnalysis` object.

**Type**
Read/Write String

**Example**
This example performs an analysis using the given method and data file:

```python
DataAnalysis.PropMethodName = _
    "C:\EZChrom Elite\Method001.met"
DataAnalysis.PropDataName = _
    "C:\EZChrom Elite\Data001.dat"
DataAnalysis.PropSampleID = "Test Sample"
DataAnalysis.SubmitAnalysis(ezAnalysisSpecifiedOptions)```
**SetDefaultAnalysisProperties Method**

Sets all of the properties for an analysis back to their default settings. This is a member of the `DataAnalysis` object.

The default settings for the data analysis properties are as follows:

<table>
<thead>
<tr>
<th>Data Analysis Property</th>
<th>Default Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropAverageCalibrationReplicates</td>
<td>False</td>
</tr>
<tr>
<td>PropCalibrationLevel</td>
<td>0</td>
</tr>
<tr>
<td>PropClearAllCalibration</td>
<td>False</td>
</tr>
<tr>
<td>PropClearCalibrationLevel</td>
<td>False</td>
</tr>
<tr>
<td>PropClearCalibrationReplicates</td>
<td>False</td>
</tr>
<tr>
<td>PropDataName</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>PropISTDAmount</td>
<td>1.0</td>
</tr>
<tr>
<td>PropMethodName</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>PropMultiplicationFactor</td>
<td>1.0</td>
</tr>
<tr>
<td>PropPrintCalibrationReport</td>
<td>False</td>
</tr>
<tr>
<td>PropPrintCustomMethodReport</td>
<td>False</td>
</tr>
<tr>
<td>PropSampleAmount</td>
<td>1.0</td>
</tr>
<tr>
<td>PropSampleID</td>
<td>&quot;&quot;</td>
</tr>
</tbody>
</table>

**Returns**

Void

**Syntax**

```
SetDefaultAnalysisProperties()
```

**Example**

This example performs an analysis using the given method and data file:

```csharp
DataAnalysis.SetDefaultAnalysisProperties
DataAnalysis.PropMethodName = _
    "C:\EZChrom Elite\Method001.met"
DataAnalysis.PropDataName = _
    "C:\EZChrom Elite\Data001.dat"
DataAnalysis.SubmitAnalysis(ezAnalysisSpecifiedOptions)
```

**SubmitAnalysisEx Method**

Submits an analysis to the run queue of the connected instrument. All of the analysis properties are used to control how the analysis is made. This is a member of the `DataAnalysis` object.

**Returns**

Read Only Long (4 byte integer value)

The return value is the run item id. This value can be used to check the run queue to get the status of the analysis. A value of -1 means the analysis was not in the run queue long enough to return a value.
Syntax
SubmitAnalysisEx(AnalysisType)

AnalysisType  Optional Long (4 byte integer value). This is a flag, from the
EZAnalysisTypeFlags enumeration, that specifies whether to analyze
the current data and method or to analyze a specified data file and
method. If this value is not specified a default value of
ezAnalysisCurrent will be used.

The EZAnalysisTypeFlags enumeration contains the following constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ezAnalysisCurrent</td>
<td>Analyzes the current data file with the current</td>
</tr>
<tr>
<td></td>
<td>method. No option settings from this object will</td>
</tr>
<tr>
<td></td>
<td>be used with this flag.</td>
</tr>
<tr>
<td>ezAnalysisSpecifiedOptions</td>
<td>Analyze the specified data file using the</td>
</tr>
<tr>
<td></td>
<td>specified method and options.</td>
</tr>
</tbody>
</table>

Example
This example performs an analysis using the given method and data file:

```vba
Dim lngRunItemID as Long

DataAnalysis.SetDefaultRunProperties
DataAnalysis.PropMethodName = "C:\EZChrom Elite\Method001.met"
DataAnalysis.PropDataName = "C:\EZChrom Elite\Data001.dat"
lngRunItemID = DataAnalysis.SubmitAnalysisEx(ezAnalysisSpecifiedOptions)
```

SubmitAnalysis Method
Submits an analysis to the run queue of the connected instrument. All of the analysis properties are
used to control how the analysis is made. This is a member of the DataAnalysis object.

Returns
Void

Syntax
SubmitAnalysis(AnalysisType)

AnalysisType  Optional Long (4 byte integer value). This is a flag, from the
EZAnalysisTypeFlags enumeration, that specifies whether to analyze
the current data and method or to analyze a specified data file and
method. If this value is not specified a default value of
ezAnalysisCurrent will be used.
The **EZAnalysisTypeFlags** enumeration contains the following constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ezAnalysisCurrent</td>
<td>Analyzes the current data file with the current method. No option settings from this object will be used with this flag.</td>
</tr>
<tr>
<td>ezAnalysisSpecifiedOptions</td>
<td>Analyze the specified data file using the specified method and options.</td>
</tr>
</tbody>
</table>

**Example**

This example performs an analysis using the given method and data file:

```vbnet
DataAnalysis.SetDefaultRunProperties
DataAnalysis.PropMethodName = "C:\EZChrom Elite\Method001.met"
DataAnalysis.PropDataName = "C:\EZChrom Elite\Data001.dat"
DataAnalysis.SubmitAnalysis(ezAnalysisSpecifiedOptions)
```
CurrentSystemInfo Object

This object contains the methods and properties needed to obtain current system information. The Instrument object returns a pointer to this object. For an overview of all objects, see the Object Hierarchy.

DataFileName Property

Returns the current data file name. This is a member of the CurrentSystemInfo object.

Type
Read Only String

Example
This example displays a message box with the name of the currently loaded data file to the user:

```vbnet
strMsg = "The current data file is " & _
        CurrentSystemInfo.DataFileName
MsgBox (strMsg)
```

DilutorFactor1 Property

Returns the dilutor factor #1 from the last analysis. This is a member of the CurrentSystemInfo object.

Type
Read Only Double (8 byte floating point value)

Example
This example displays dilutor factor #1 from the last analysis of the data file:

```vbnet
strMsg = "The dilutor factor #1 is " & _
        CurrentSystemInfo.DilutorFactor1
MsgBox (strMsg)
```

DilutorFactor2 Property

Returns the dilutor factor #2 from the last analysis. This is a member of the CurrentSystemInfo object.

Type
Read Only Double (8 byte floating point value)
Example
This example displays dilutor factor #2 from the last analysis of the data file:

    strMsg = "The dilutor factor #2 is " & _
            CurrentSystemInfo.DilutorFactor2
    MsgBox (strMsg)

DilutorFactor3 Property
Returns the dilutor factor #3 from the last analysis. This is a member of the CurrentSystemInfo object.

Type
Read Only Double (8 byte floating point value)

Example
This example displays dilutor factor #3 from the last analysis of the data file:

    strMsg = "The dilutor factor #3 is " & _
            CurrentSystemInfo.DilutorFactor3
    MsgBox (strMsg)

InstrumentName Property
Returns the current instrument name. This is a member of the CurrentSystemInfo object.

Type
Read Only String

Example
This example displays a message box with the name of the current instrument to the user:

    strMsg = "The current instrument is " & _
            CurrentSystemInfo.InstrumentName
    MsgBox (strMsg)

ISTDAmount Property
Returns the current ISTD amount. This is a member of the CurrentSystemInfo object.

Type
Read Only Double (8 byte floating point value)
Example
This example displays a message box with the current ISTD amount to the user:

```vbscript
strMsg = "The current ISTD amount is " & _
        CurrentSystemInfo.ISTDAmount
MsgBox (strMsg)
```

**MethodFileName Property**

Returns the current method file name. This is a member of the `CurrentSystemInfo` object.

**Type**

Read Only String

**Example**

This example displays a message box with the name of the currently loaded method file to the user:

```vbscript
strMsg = "The current method file is " & _
        CurrentSystemInfo.MethodFileName
MsgBox (strMsg)
```

**MultiplierFactor Property**

Returns the total multiplier factor from the last analysis. This is a member of the `CurrentSystemInfo` object.

**Type**

Read Only Double (8 byte floating point value)

**Example**

This example displays total multiplier factor from the last analysis of the data file:

```vbscript
strMsg = "The total multiplier factor is " & _
        CurrentSystemInfo.MultiplierFactor
MsgBox (strMsg)
```

**MultiplierFactor1 Property**

Returns the multiplier factor #1 from the last analysis. This is a member of the `CurrentSystemInfo` object.

**Type**

Read Only Double (8 byte floating point value)

**Example**
This example displays multiplier factor #1 from the last analysis of the data file:

```vba
strMsg = "The multiplier factor #1 is " &
        CurrentSystemInfo.MultiplierFactor1
MsgBox (strMsg)
```

### MultiplierFactor2 Property

Returns the multiplier factor #2 from the last analysis. This is a member of the `CurrentSystemInfo` object.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays multiplier factor #2 from the last analysis of the data file:

```vba
strMsg = "The multiplier factor #2 is " &
        CurrentSystemInfo.MultiplierFactor2
MsgBox (strMsg)
```

### MultiplierFactor3 Property

Returns the multiplier factor #3 from the last analysis. This is a member of the `CurrentSystemInfo` object.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays multiplier factor from the last analysis of the data file:

```vba
strMsg = "The multiplier factor #3 is " &
        CurrentSystemInfo.MultiplierFactor3
MsgBox (strMsg)
```

### Name Property

Returns the name of this object. This is a member of the `CurrentSystemInfo` object.

**Type**
Read Only String

**Example**
This example displays a message box with the name of this object to the user:
strMsg = "The name of the CurrentSystemInfo object is " & _
CurrentSystemInfo.Name
MsgBox (strMsg)

**OEM_ID Property**

Returns the OEM ID. This is a member of the CurrentSystemInfo object.

**Type**
Read Only Long (4 byte integer value)

**Example**
This example displays the OEM ID to the user:

```
MsgBox ("The OEM ID is: " & _ CurrentSystemInfo.OEM_ID)
```

**PretreatFileName Property**

Returns the current pretreat file name. This is a member of the CurrentSystemInfo object.

**Type**
Read Only String

**Example**
This example displays a message box with the name of the currently loaded pretreat file to the user:

```
strMsg = "The current pretreat file is " & _
CurrentSystemInfo.PretreatFileName
MsgBox (strMsg)
```

**ProductName Property**

Returns the product name. This is a member of the CurrentSystemInfo object.

**Type**
Read Only String

**Example**
This example displays the product name to the user:

```
MsgBox ("The product name is: " & CurrentSystemInfo.ProductName)
```

**SampleAmount Property**

Returns the current sample amount. This is a member of the CurrentSystemInfo object.
Type
Read Only Double (8 byte floating point value)

Example
This example displays a message box with the current sample amount to the user:

```
strMsg = "The current sample amount is " & _
         CurrentSystemInfo.SampleAmount
MsgBox (strMsg)
```

**SampleID Property**

Returns the current sample ID. This is a member of the [CurrentSystemInfo](#) object.

Type
Read Only String

Example
This example displays a message box with the current sample ID to the user:

```
strMsg = "The current sample ID is " & _
         CurrentSystemInfo.SampleID
MsgBox (strMsg)
```

**SequenceFileName Property**

Returns the current sequence file name. This is a member of the [CurrentSystemInfo](#) object.

Type
Read Only String

Example
This example displays a message box with the name of the currently loaded sequence file to the user:

```
strMsg = "The current sequence file name is " & _
         CurrentSystemInfoSEQUENCEFILENAME
MsgBox (strMsg)
```

**SoftwareBuild Property**

Returns the software build information. This is a member of the [CurrentSystemInfo](#) object.

Type
Read Only String

Example
This example displays the software build information to the user:

```vba
MsgBox ("The software build number is: " & _
CurrentSystemInfo.SoftwareBuild)
```

**SoftwareVersion Property**

Returns the current software version. This is a member of the `CurrentSystemInfo` object.

**Type**
Read Only String

**Example**
This example displays a message box with the current software version to the user:

```vba
strMsg = "The current software version is " & _
CurrentSystemInfo.SoftwareVersion
MsgBox (strMsg)
```

**UserName Property**

Returns the name of the user that is currently logged into the instrument. This is a member of the `CurrentSystemInfo` object.

**Type**
Read Only String

**Example**
This example displays a message box with the name of the current user:

```vba
strMsg = "The name of the current user is " & _
CurrentSystemInfo.UserName
MsgBox (strMsg)
```
DataFileInfo Object

This object contains the methods and properties needed to obtain information about a loaded data file. The Instrument object returns a pointer to this object. For an overview of all objects, see the Object Hierarchy.

**AcquisitionDateTime Property**

Returns the date and time this data file was acquired. This is a member of the DataFileInfo object.

**Type**
Read Only String

**Example**
This example displays a message box with the data file date and time of acquisition:

```vbnet
strMsg = "The data file was acquired on " & _
DataFileInfo.AcquisitionDateTime
MsgBox (strMsg)
```

**AnalysisComputerName Property**

Returns the data analysis computer name. This is a member of the DataFileInfo object.

**Type**
Read Only String

**Example**
This example displays a message box with the computer name of the last analysis of this data file:

```vbnet
strMsg = "The current data file analysis computer name is " & _
DataFileInfo.AnalysisComputerName
MsgBox (strMsg)
```

**AnalysisDateTime Property**

Returns the date and time of the last analysis of this data file. This is a member of the DataFileInfo object.

**Type**
Read Only String

**Example**
This example displays a message box with the data file date and time of analysis:
strMsg = "The data file was last analyzed on " & _
DataFileInfo.AnalysisDateTime
MsgBox (strMsg)

BCDValue Property

Returns the BCD value available at the start of the run. This is a member of the DataFileInfo object.

Type
Read Only Long (4 byte integer value)

Example
This example displays the BCD value that was available at the start of the run:

strMsg = "The BCD value at the start of the run was " & _
DataFileInfo.BCDValue
MsgBox (strMsg)

Description Property

Returns the data file description. This is a member of the DataFileInfo object.

Type
Read Only String

Example
This example displays a message box with the data file description:

strMsg = "The data file description is " & _
DataFileInfo.Description
MsgBox (strMsg)

DilutorFactor1 Property

Returns the dilutor factor #1 from the last analysis. This is a member of the DataFileInfo object.

Type
Read Only Double (8 byte floating point value)

Example
This example displays dilutor factor #1 from the last analysis of the data file:

strMsg = "The dilutor factor #1 is " & _
DataFileInfo.DilutorFactor1
MsgBox (strMsg)
**DilutorFactor2 Property**

Returns the dilutor factor #2 from the last analysis. This is a member of the [DataFileInfo](#) object.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays dilutor factor #2 from the last analysis of the data file:

```vbnet
strMsg = "The dilutor factor #2 is " & _
        DataFileInfo. DilutorFactor2
MsgBox (strMsg)
```

**DilutorFactor3 Property**

Returns the dilutor factor #3 from the last analysis. This is a member of the [DataFileInfo](#) object.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays dilutor factor #3 from the last analysis of the data file:

```vbnet
strMsg = "The dilutor factor #3 is " & _
        DataFileInfo. DilutorFactor3
MsgBox (strMsg)
```

**ESignatures Property**

Returns the [ESignatures](#) object. This is a member of the [DataFileInfo](#) object.

**Type**
Read Only Object

**Example**
This example gets the number of users who signed a data doc:

```vbnet
Dim nTotalSignatures As Integer
nTotalSignatures = DataFileInfo.ESignatures.Count
```

**GetCustomParameter Method**

Returns the value of the specified system-wide custom parameter. This is a member of the [DataFileInfo](#) object.

**Returns**
Void

Syntax
GetCustomParameter(ParamID, vParamValue)

- **ParamID**: Long (4 byte integer value). This is the parameter id of the requested parameter. Custom parameters have a parameter id in the range of 2000 to 2500. The first custom parameter defined in a method will have a value of 2000. The next custom parameter will have a value of 2001, etc.
- **vParamValue**: Variant. This is a variant that will receive the value of the specified custom parameter. The type of the variant will be either Double or String, depending on the type of the custom parameter.

Example
This example gets the value of the system-wide custom parameter with a parameter id of 2005:

```vbs
Call DataFileInfo.GetCustomParameter(2005, vParamValue)
strMsg = "The custom parameter value is " & vParamValue
MsgBox (strMsg)
```

GetAuditTrailMode Method

Gets the audit trail mode for the current data file. This is a member of the DataFileInfo object.

Returns
Long (4 byte integer value)

The return value will be one of the following:

- 0 – Prompt for reason
- 2 – Do not prompt for reason
- any other value – undefined, audit trial may not be enabled

Syntax
GetAuditTrailMode ()

Example
This example gets the audit trail mode after checking that the audit trail is enabled:

```vbs
Dim bAuditTrailEnabled as Boolean
Dim lMode as Long

bAuditTrailEnabled = DataFileInfo.IsAuditTrailEnabled
If (bAuditTrailEnabled) then
    lMode = bSuccess = DataFileInfo.GetAuditTrailMode()
Else
    ' audit trail mode is undefined
End If
```
**InstrumentName Property**

Returns the instrument name that was used to acquire this data file. This is a member of the DataFileInfo object.

**Type**
Read Only String

**Example**
This example displays a message box with the name of the instrument used to acquire this data file:

```
strMsg = "The data file instrument is " & _
        DataFileInfo.InstrumentName
MsgBox (strMsg)
```

**IsAuditTrailEnabled Method**

Returns the audit trail enabled state for the open data file. This is a member of the DataFileInfo object.

**Returns**
Boolean

**Syntax**
IsAuditTrailEnabled()

**Example**
This example determines whether the audit trail is enabled for this data file:

```
Dim bEnabled as Boolean
bEnabled = DataFileInfo.IsAuditTrailEnabled
```

**IsElectronicallySigned Method**

Returns TRUE or FALSE to indicate if the currently loaded data file has been electronically signed. This is a member of the DataFileInfo object.

**Type**
Read Only Boolean

**Syntax**
IsElectronicallySigned()

**Example**
This example determines whether or not the current data file is electronically signed:

```vba
Dim bESigned As Boolean
bESigned = DataFileInfo.IsElectronicallySigned
```

**ISTDAmount Property**

Returns the ISTD amount from the last analysis. This is a member of the `DataFileInfo` object.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays ISTD amount from the last analysis of the data file:

```vba
strMsg = "The ISTD amount is " & DataFileInfo.ISTDAmount
MsgBox (strMsg)
```

**MethodFileName Property**

Returns the name of the method file that was used to acquire this data file. This is a member of the `DataFileInfo` object.

**Type**
Read Only String

**Example**
This example displays a message box with the name of the method used to acquire the data file:

```vba
strMsg = "The method used to acquire this data file is " & _
         DataFileInfo.MethodFileName
MsgBox (strMsg)
```

**MultiplierFactor Property**

Returns the total multiplier factor from the last analysis. This is a member of the `DataFileInfo` object.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays total multiplier factor from the last analysis of the data file:

```vba
strMsg = "The total multiplier factor is " & _
```
DataFileInfo.MultiplierFactor
MsgBox (strMsg)

**MultiplierFactor1 Property**
Returns the multiplier factor #1 from the last analysis. This is a member of the DataFileInfo object.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays multiplier factor #1 from the last analysis of the data file:

```vba
strMsg = "The multiplier factor #1 is " & _
    DataFileInfo.MultiplierFactor1
MsgBox (strMsg)
```

**MultiplierFactor2 Property**
Returns the multiplier factor #2 from the last analysis. This is a member of the DataFileInfo object.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays multiplier factor #2 from the last analysis of the data file:

```vba
strMsg = "The multiplier factor #2 is " & _
    DataFileInfo.MultiplierFactor2
MsgBox (strMsg)
```

**MultiplierFactor3 Property**
Returns the multiplier factor #3 from the last analysis. This is a member of the DataFileInfo object.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays multiplier factor from the last analysis of the data file:

```vba
strMsg = "The multiplier factor #3 is " & _
    DataFileInfo.MultiplierFactor3
MsgBox (strMsg)
```
**Name Property**

Returns the name of this object. This is a member of the `DataFileInfo` object.

**Type**
Read Only String

**Example**
This example displays a message box with the name of this object to the user:

```vba
strMsg = "The name of the DataFileInfo object is " & _
        DataFileInfo.Name
MsgBox (strMsg)
```

**PDAFromWavelength Property**

Returns the Start/From wavelength of a PDA spectrum. If a PDA data file is not loaded, the value will be 0. This is a member of the `DataFileInfo` object.

**Type**
Read Only Long

**Example**
This example displays the from wavelength of the PDA data:

```vba
strMsg = "The from wavelength is " & _
        DataFileInfo.PDAFromWavelength
MsgBox (strMsg)
```

**PDAToWavelength Property**

Returns the End/To wavelength of a PDA spectrum. If a PDA data file is not loaded, the value will be 0. This is a member of the `DataFileInfo` object.

**Type**
Read Only Long

**Example**
This example displays the to wavelength of the PDA data:

```vba
strMsg = "The to wavelength is " & _
        DataFileInfo.PDAToWavelength
MsgBox (strMsg)
```
**PDAWaveInterval Property**

Returns the Interval between wavelengths of a PDA spectrum. If a PDA data file is not loaded, the value will be 1. This is a member of the DataFileInfo object.

**Type**
Read Only Long

**Example**
This example displays the wavelength interval of the PDA data:

```vbscript
strMsg = "The wavelength interval is " & _
        DataFileInfo.PDAWaveInterval
MsgBox (strMsg)
```

**ReadOnly Property**

Returns TRUE or FALSE to indicate if the currently loaded data file is read-only. A data file will be read-only if another client has already opened this same data file. This is a member of the DataFileInfo object.

**Type**
Read Only Boolean

**Example**
This example determines whether or not the current data file is read-only:

```vbscript
bReadOnly = DataFileInfo.ReadOnly
If (bReadOnly = True) Then
    MsgBox ("The current data file is read-only")
Else
    MsgBox ("The current data file is not read-only")
End If
```

**SampleAmount Property**

Returns the sample amount from the last analysis. This is a member of the DataFileInfo object.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays sample amount from the last analysis of the data file:

```vbscript
strMsg = "The sample amount is " & DataFileInfo.SampleAmount
MsgBox (strMsg)
```
**SampleID Property**

Returns the sample ID of the data. This is a member of the `DataFileInfo` object.

**Type**
Read Only String

**Example**
This example displays a message box with the sample ID from the data file:

```
strMsg = "The sample ID is " & DataFileInfo.SampleID
MsgBox (strMsg)
```

**SetAuditTrailMode Method**

Sets the audit trail mode for the current data file. This is a member of the `DataFileInfo` object.

**Returns**
Boolean

**Syntax**

```
SetAuditTrailMode (lMode)
```

**lMode**
Long (4 byte integer value). This is the audit trail mode to be set. Only a value of 2 is valid at this time.

2 – Do not prompt for reason

**Example**
This example sets the audit trail mode for no prompting:

```
Dim bSuccess as Boolean
bSuccess = DataFileInfo.SetAuditTrailMode(2)
```

**SoftwareVersion Property**

Returns the version of the software that the data file was acquired with. This is a member of the `DataFileInfo` object.

**Type**
Read Only String

**Example**
This example displays a message box with the software version that the data file was acquired with:

```
strMsg = "The software version is " & _
        DataFileInfo.SoftwareVersion
MsgBox (strMsg)
```
**UserName Property**

Returns the name of the user that acquired this data file. This is a member of the `DataFileInfo` object.

**Type**
Read Only String

**Example**
This example displays a message box with the name of the user that acquired the data file:

```vbnet
strMsg = "The user that acquired this data file is " & _
        DataFileInfo.UserName
MsgBox (strMsg)
```

**Vial Property**

Returns the vial that was used to acquire the sample. This is a member of the `DataFileInfo` object.

**Type**
Read Only Long (4 byte integer value)

**Example**
This example displays a message box with the vial that was used to acquire the sample:

```vbnet
strMsg = "The vial used for this sample was " & _
        DataFileInfo.Vial
MsgBox (strMsg)
```

**VialText Property**

Returns the vial that will be used for the first injection of a sample in a single run by the `SubmitSingleRun()` method. This is a member of the `DataFileInfo` object.

An empty string indicates that no vial was used.

Vials are specified using the formatted, decoded string of the autosampler user interface. This vial string corresponds to the encoded numeric vial identifier used internally by Elite. In order to work with vial identifiers using a numeric format, use `PropInitialVial` instead.

**Type**
Read-only String

**Example**
This example displays a message box with the vial that was used to acquire the sample:

```vbnet
strMsg = "The vial used for this sample was " & _
        DataFileInfo.VialText
MsgBox (strMsg)
```
**Volume Property**

Returns the volume of the sample that was injected. This is a member of the `DataFileInfo` object.

**Type**
Read Only Single (4 byte floating point value)

**Example**
This example displays a message box with the volume that was injected:

```vbnet
strMsg = "The volume injected was " & _
        DataFileInfo.Volume
MsgBox (strMsg)
```
RunQueue Object

This object contains a collection of all the RunQueueItem objects that are currently available. Applications can enumerate through the collection of run queue items by using For … Next statements or by using For … Each statements. For an overview of all objects, see the Object Hierarchy.

NOTE:
A call to either the Instrument RefreshRunQueue method or the RunQueue RefreshItemList method should be made before calling any other RunQueue methods or properties since the run queue changes as runs are added, completed, or aborted.

Example
This example displays the status of each run queue item in the collection using For … Each statements:

```
For Each RunQueueItem In RunQueue
    MsgBox ("Run Queue Item Status: " & RunQueueItem.ItemStatus)
Next
```

This example displays the status of each run queue item in the collection using For … Next statements:

```
nCount = RunQueue.Count
For nIndex = 0 To nCount - 1
    MsgBox ("Run Queue Item Status: " & RunQueue(nIndex).ItemStatus)
Next nIndex
```

AbortMultipleRuns Method

Removes multiple items from the run queue of the connected instrument. If an item is currently running it will be aborted before being removed. This is a member of the RunQueue object.

Returns
Void

Syntax
AbortMultipleRuns(AbortFlag)

```
AbortFlag  Optional Long (4 byte integer value). This is a constant from the EZAbortRunFlags enumeration. This parameter has a default value of ezAbortAllClientRuns if it is not specified.
```

The AbortFlag is one of the following EZAbortRunFlags constants:

```
<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ezAbortAllClientRuns</td>
<td>Aborts all runs in the run queue that were submitted by this user, including any runs started in previous sessions.</td>
</tr>
<tr>
<td>ezAbortAllRuns</td>
<td>Currently, if the first run in the queue belongs to the user requesting the abort, all runs in the queue will be aborted. If the first run in the queue does not belong to the user, a check will be made to determine if the user has System or Instrument Administration rights. If the user does</td>
</tr>
</tbody>
</table>
```
have these rights, all runs in the queue will be aborted. If the user does not have these rights, all runs will remain in the queue.

Example
This example removes all items from the run queue that were submitted by this client:

```vba
If RunQueue.Count > 0 Then
    RunQueue.AbortMultipleRuns(ezAbortAllClientRuns)
End If
```

**Count Property**

Returns the number of run queue items. It is recommended that the user call Instrument.RefreshRunQueue first to get an accurate count. This is a member of the RunQueue object.

**Type**
Read Only Long (4 byte integer value)

**Syntax**
Count(ClientFlag)

**ClientFlag**
Optional Long (4 byte integer value). If not specified, then this parameter will have a default value of **ezCurrentSession**. This is a flag that specifies the type of run queue items to be counted.

The ClientFlag is one of the following **EZClientTypeFlags** constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ezClientAll</td>
<td>This flag will cause all run queue items to be counted, even if the items were not submitted by this session or user.</td>
</tr>
<tr>
<td>ezClientCurrent</td>
<td>This flag is now obsolete. Please use the ezCurrentSession flag instead.</td>
</tr>
<tr>
<td>ezCurrentSession</td>
<td>This flag will cause only the run queue items submitted by this session of automation to be counted.</td>
</tr>
<tr>
<td>ezCurrentUser</td>
<td>This flag will cause only the run queue items submitted by this user name to be counted, no matter what session they were submitted in.</td>
</tr>
</tbody>
</table>

**Example**
This example displays the status of each run queue item to the user:

```vba
nCount = RunQueue.Count(ezClientAll)
For nIndex = 0 To nCount - 1
    MsgBox ("Run Queue Item Status: " & 
            RunQueue(nIndex).ItemStatus)
Next nIndex
```
**Item Property**

Returns the RunQueueItem object at the given indexed location. This is a member of the RunQueue object.

**Type**
Read Only Object

**Syntax**
Item(Index)

**Index** Required Long (4 byte integer value). This is a zero based index that specifies the Run Queue Item object to be returned.

**Example**
This example displays the status of each run queue item in the collection:

```vbnet
nCount = RunQueue.Count
For nIndex = 0 To nCount - 1
    MsgBox ("Run Queue Item Status: " & RunQueue.Item(nIndex).ItemStatus)
Next nIndex
```

**Remarks**
Item is the default member of this collection, so the following two lines of code are equivalent:

```vbnet
RunQueue.Item(1)
RunQueue(1)
```

**Name Property**

Returns the name of this object. This is a member of the RunQueue object.

**Type**
Read Only String

**Example**
This example displays a message box with the name of this object to the user:

```vbnet
strMsg = "The name of the RunQueue object is " & RunQueue.Name
MsgBox (strMsg)
```
**RefreshItemList Method**

Refreshes the list of available run queue items. This is a member of the RunQueue object.

**Returns**
Void

**Syntax**
RefreshItemList()

**Example**
This example removes the currently running item from the run queue:

```vba
RunQueue.RefreshItemList
If RunQueue.Count > 0 Then
    RunQueue.Remove(0)
End If
```

**Remove Method**

Removes an item from the run queue of the connected instrument. If the item is currently running, then it will be aborted before being removed. This is a member of the RunQueue object.

**Returns**
Void

**Syntax**
Remove(Index)

**Index**
Required Long (4 byte integer value). This is a zero based index that specifies the Run Queue Item object to be removed.

**Example**
This example removes the currently running item from the run queue:

```vba
If RunQueue.Count > 0 Then
    RunQueue.Remove(0)
End If
```
RunQueueItem Object

This object represents an item in the run queue of an instrument that is either pending or being processed. The RunQueue collection object contains all of the run queue item objects that are currently available. For an overview of all objects, see the Object Hierarchy.

NOTE:
A call to either the Instrument RefreshRunQueue method or the RunQueue RefreshItemList method should be made before calling any RunQueueItem methods or properties since the run queue changes as runs are added, completed, or aborted.

IsCurrentClient Method

NOTE: This method is now obsolete. The word "client" did not seem to have a clear meaning. Therefore two new methods have been added to avoid confusion. This method will call the IsCurrentSession method, which is the replacement for this method. Please start using the IsCurrentSession method instead of this one wherever possible.

Returns True or False to indicate if this run queue item was submitted by the current session. This is a member of the RunQueueItem object.

Returns
Boolean

Syntax
IsCurrentClient()

Example
This example determines whether or not this run queue item was submitted by the current client, and displays a message box to the user:

```vbnet
bCurrentClient = RunQueueItem.IsCurrentClient
If (bCurrentClient = True) Then
    MsgBox ("Item submitted by client")
Else
    MsgBox ("Item NOT submitted by client")
End If
```

IsCurrentSession Method

Returns True or False to indicate if this run queue item was submitted by the current session of automation. This is a member of the RunQueueItem object.

Returns
Boolean
Syntax
IsCurrentSession()

Example
This example determines whether or not this run queue item was submitted by the current automation session, and displays a message box to the user:

```vba
bCurrentClient = RunQueueItem.IsCurrentSession
If (bCurrentClient = True) Then
    MsgBox ("Item submitted by this session")
Else
    MsgBox ("Item NOT submitted by this session")
End If
```

IsCurrentUser Method

Returns True or False to indicate if this run queue item was submitted by the current user. This is a member of the RunQueueItem object.

Returns
Boolean

Syntax
IsCurrentUser()

Example
This example determines whether or not this run queue item was submitted by the current user name, and displays a message box to the user:

```vba
bCurrentClient = RunQueueItem.IsCurrentUser
If (bCurrentClient = True) Then
    MsgBox ("Item submitted by current user")
Else
    MsgBox ("Item NOT submitted by current user")
End If
```

ItemDataName Property

Returns the data file name of this run queue item, if the item is a single run type. This is a member of the RunQueueItem object.

Type
Read Only String

Example
This example displays the data file name of this run queue item:

```vba
If (RunQueueItem.ItemType = ezItemTypeSingleRun) Then
    MsgBox ("Data File Name: " & _
            RunQueueItem.ItemDataName)
End If
```
**ItemDescription Property**

Returns the description of this run queue item. This is a member of the `RunQueueItem` object.

**Type**
Read Only String

**Example**
This example displays the description of this run queue item:

```vbscript
MsgBox ("Description: " & RunQueueItem.ItemDescription)
```

**ItemID Property**

Returns the internal ID of this run queue item. This is a member of the `RunQueueItem` object.

**Type**
Read Only Long (4 byte integer value)

**Example**
This example displays the internal ID of this run queue item:

```vbscript
MsgBox ("Internal Item ID: " & RunQueueItem.ItemID)
```

**ItemMethodName Property**

Returns the method name of this run queue item, if the item is a single run type. This is a member of the `RunQueueItem` object.

**Type**
Read Only String

**Example**
This example displays the method name of this run queue item:

```vbscript
If (RunQueueItem.ItemType = ezItemTypeSingleRun) Then
    MsgBox ("Method Name: " & RunQueueItem.ItemMethodName)
End If
```
**ItemPriorityFlag Property**

Returns True or False to indicate if this run queue item is a priority item. This is a member of the `RunQueueItem` object.

**Type**
Read Only Boolean

**Example**
This example displays a message box if this run queue item is a priority item:

```vbnet
If (RunQueueItem.ItemPriorityFlag = True) Then
    MsgBox ("Run Queue Item is a priority item")
End If
```

**ItemSequenceName Property**

Returns the sequence name of this run queue item, if the item is a sequence type. This is a member of the `RunQueueItem` object.

**Type**
Read Only String

**Example**
This example displays the sequence name of this run queue item:

```vbnet
If (RunQueueItem.ItemType = ezItemTypeSequenceRun) Then
    MsgBox ("Sequence Name: " & RunQueueItem.ItemSequenceName)
End If
```

**ItemStatus Property**

Returns the status of this run queue item. This is a member of the `RunQueueItem` object.

**Type**
Read Only Long (4 byte integer value)

The value returned is one of the following `EZRunQueueItemStatusFlags` constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ezItemStatusPending</td>
<td>This run queue item is pending.</td>
</tr>
<tr>
<td>ezItemStatusProcessing</td>
<td>This run queue item is being processed.</td>
</tr>
<tr>
<td>ezItemStatusSuspended</td>
<td>This run queue item is suspended.</td>
</tr>
</tbody>
</table>

**Example**
This example displays the status of this run queue item:
If (RunQueueItem.ItemStatus = ezItemStatusPending) Then
    MsgBox("Item Status: Pending")
ElseIf (RunQueueItem.ItemStatus = ezItemStatusProcessing) Then
    MsgBox("Item Status: Processing")
ElseIf (RunQueueItem.ItemStatus = ezItemStatusSuspended) Then
    MsgBox("Item Status: Suspended")
End If

**ItemType Property**

Returns the type of this run queue item. This is a member of the RunQueueItem object.

*Type*
Read Only Long (4 byte integer value)

The value returned is one of the following EZRunQueueItemFlags constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ezItemTypeSingleRun</td>
<td>This run queue item is a single run.</td>
</tr>
<tr>
<td>ezItemTypeSingleAnalysis</td>
<td>This run queue item is a single analysis.</td>
</tr>
<tr>
<td>ezItemTypeSequenceRun</td>
<td>This run queue item is a sequence run.</td>
</tr>
<tr>
<td>ezItemTypeSequenceAnalysis</td>
<td>This run queue item is a sequence analysis.</td>
</tr>
<tr>
<td>ezItemTypePreviewRun</td>
<td>This run queue item is a preview run.</td>
</tr>
</tbody>
</table>

**Example**
This example displays a message box if this item is a single run:

```vba
If (RunQueueItem.ItemType = ezItemTypeSingleRun) Then
    MsgBox("Run Queue Item is a Single Run")
End If
```

**ItemUser Property**

Returns the name of the user that submitted this run queue item. This is a member of the RunQueueItem object.

*Type*
Read Only String

**Example**
This example displays the name of the user that submitted this run queue item:

```vba
MsgBox("User: " & RunQueueItem.ItemUser)
```
**Name Property**

Returns the name of this run queue item. This is a member of the RunQueueItem object.

**Type**
Read Only String

**Example**
This example displays a message box with the name of this run queue item to the user:

```vba
strMsg = "The name of this run queue item is " & _
        RunQueueItem.Name
MsgBox (strMsg)
```
**Devices Object**

This object contains a collection of all the Device objects that are currently available. Applications can enumerate through the collection of devices by using For … Next statements or by using For … Each statements. For an overview of all objects, see the Object Hierarchy.

**Example**

This example displays the name of each device in the collection using For … Each statements:

```vba
For Each Device In Devices
    MsgBox("Device Name: " & Device.Name)
Next
```

This example displays the name of each device in the collection using For … Next statements:

```vba
nCount = Devices.Count
For nIndex = 0 To nCount - 1
    MsgBox("Device Name: " & Devices(nIndex).Name)
Next nIndex
```

**Count Property**

Returns the number of devices available. This is a member of the Devices object.

**Type**

Read Only Long (4 byte integer value)

**Example**

This example displays the name of each device to the user:

```vba
nCount = Devices.Count
For nIndex = 0 To nCount - 1
    MsgBox("Device Name: " & Devices(nIndex).Name)
Next nIndex
```

**Item Property**

Returns the Device object at the given indexed location. This is a member of the Devices object.

**Type**

Read Only Object

**Syntax**
**Item(Index)**

**Index**
Required Long (4 byte integer value). This is a zero based index that specifies the Device object to be returned.

**Example**
This example displays the name of each device in the collection:

```vba
nCount = Devices.Count
For nIndex = 0 To nCount - 1
    MsgBox ("Device Name: " & Devices.Item(nIndex).Name)
Next nIndex
```

**Remarks**
Item is the default member of this collection, so the following two lines of code are equivalent:

```vba
Devices.Item(1)
Devices(1)
```

**Name Property**

Returns the name of this object. This is a member of the Devices object.

**Type**
Read Only String

**Example**
This example displays a message box with the name of this object to the user:

```vba
strMsg = "The name of the Devices object is " & _
         Devices.Name
MsgBox (strMsg)
```
**Device Object**

This object represents a device that is configured for the current instrument. The Devices collection object contains all of the device objects that are currently available. For an overview of all objects, see the Object Hierarchy.

**Category Property**

Returns the category type of this device. The category type is a classification of the device. This is a member of the Device object.

*Note: The category type is determined by the individual device. See the device specific documentation for the device of interest.*

**Type**

Read Only Long (4 byte integer value)

The value returned is one of the following EZDeviceCategoryFlags constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ezDeviceCategoryNotSpecified</td>
<td>The device category is not specified.</td>
</tr>
<tr>
<td>ezDeviceCategoryAutosampler</td>
<td>The device is an autosampler.</td>
</tr>
<tr>
<td>ezDeviceCategoryDetector</td>
<td>The device is a detector.</td>
</tr>
<tr>
<td>ezDeviceCategoryPump</td>
<td>The device is a pump.</td>
</tr>
<tr>
<td>ezDeviceCategorySystemController</td>
<td>The device is a system controller.</td>
</tr>
<tr>
<td>ezDeviceCategoryFractionCollector</td>
<td>The device is a fraction collector.</td>
</tr>
<tr>
<td>ezDeviceCategoryAcquisitionSource</td>
<td>The device is an acquisition source.</td>
</tr>
<tr>
<td>ezDeviceCategoryValve</td>
<td>The device is a valve.</td>
</tr>
<tr>
<td>ezDeviceCategoryPDA</td>
<td>The device is a PDA detector.</td>
</tr>
<tr>
<td>ezDeviceCategoryOven</td>
<td>The device is an oven.</td>
</tr>
<tr>
<td>ezDeviceCategoryContacts</td>
<td>The device is a contact closure device.</td>
</tr>
<tr>
<td>ezDeviceCategoryMisc</td>
<td>The device is a miscellaneous device.</td>
</tr>
<tr>
<td>ezDeviceCategoryGC</td>
<td>The device is a GC.</td>
</tr>
<tr>
<td>ezDeviceCategoryADBoard</td>
<td>The device is an analog to digital board.</td>
</tr>
<tr>
<td>ezDeviceCategorySubController</td>
<td>The device is a sub-controller.</td>
</tr>
<tr>
<td>ezDeviceCategoryCooler</td>
<td>The device is a cooler.</td>
</tr>
</tbody>
</table>

**Example**

This example displays a message box if this device is a pump:

```vbnet
If (Device.Category = ezDeviceCategoryPump) Then
    MsgBox ("This device is a pump")
End If
```
**Channel_ID Property**

Returns the channel ID associated with this device. This is a member of the Device object.

**Type**
Read Only Long (4 byte integer value)

**Example**
This example displays a message box with the channel ID associated with the device:

```vbnet
strMsg = "The channel ID for the current " & Device.Channel_ID
MsgBox (strMsg)
```

**GetDeviceStatus Method**

Gets current status variant for a device. This is a member of the Device object.

*Note:* The individual device determines the format of status variant. See the device specific documentation for the device of interest. This command will only work with Rapid Control Devices that have implemented the GetAutomationDeviceInfo method in their ConfigIF.

**Returns**
Long (4 byte integer value). The value returned is one of the following constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_OK (0)</td>
<td>Status variant was successfully retrieved.</td>
</tr>
<tr>
<td>E_FAIL (a large negative number)</td>
<td>Failed to retrieve status variant. This error code would also be returned if the instrument is opened offline.</td>
</tr>
</tbody>
</table>

**Syntax**

GetDeviceStatus (Status)

**Command**
Required Variant pointer. This is the variant in which the status information would be filled and returned. The format of this variant is determined by the individual device. See the device specific documentation for the device of interest.

**Example**
This example gets the status variant from a device:

```vbnet
Dim vStatus as Variant
Dim resultErrorCode as Long

resultErrorCode = GetDeviceStatus (vStatus)
ProcessStatus(resultErrorCode,vStatus)
```

**InterfaceVersionNum Property**

Returns the interface version number for this device interface for the instrument setup command. This is a member of the Device object.
**Type**
Read Only Long (4 byte integer value)

**Example**
This example displays a message box with the interface version number for the device:

```vba
strMsg = "The interface version for the current " & "device is " & Device.InterfaceVersionNum
MsgBox (strMsg)
```

**Name Property**
Returns the name of this device. This is a member of the Device object.

*Note: This name is determined by the individual device. See the device specific documentation for the device of interest.*

**Type**
Read Only String

**Example**
This example displays the name of this device to the user:

```vba
MsgBox ("Device Name: " & Device.Name)
```

**SendCommand Method**
Sends a command to a device. This is a member of the Device object.

*Note: The available device commands are determined by the individual device. See the device specific documentation for the device of interest.*

**Returns**
Void

**Syntax**
SendCommand(CommandCode, CommandString)

- **CommandCode**
  Required Long (4 byte integer value). This is command code that will be sent to the device. This command code is determined by the individual device. See the device specific documentation for the device of interest.

- **CommandString**
  Required String. This is command string that will be sent to the device. A response from the device may also be sent back in this string. The format of this command string is determined by the individual device. See the device specific documentation for the device of interest.

**Example**
This example sends a command to a device and then displays the response to the user:

```vba
strDeviceCommand = "GetStatus"
Call SendCommand(1, strDeviceCommand)
MsgBox ("Device Response: " & strDeviceCommand)
```

### SendCommandEx Method

Sends a command to a device. This is a member of the **Device** object.

*Note: The available device commands are determined by the individual device. See the device specific documentation for the device of interest.*

**Returns**

Long (4 byte integer value)

The value returned will depend on the error codes the user sets up for this type of command. The default error code setting is E_FAIL (a large negative number). It is recommended that S_OK (0) be returned if the command was executed properly.

**Syntax**

SendCommandEx(CommandCode, Command)

- **CommandCode**
  Required Long (4 byte integer value). This is command code that will be sent to the device. This command code is determined by the individual device. See the device specific documentation for the device of interest.

- **Command**
  Required Variant pointer. This is the command variant that will be sent to the device. A response from the device may also be sent back in this variant. The format of this command variant is determined by the individual device. See the device specific documentation for the device of interest.

**Example**

This example sends a command to a device:

```vba
Dim vCommand as Variant
Dim resultErrorCode as Long

SetCommand(vCommand); // Sets information
resultErrorCode = SendCommandEx(1, vCommand)
ProcessResponse(vCommand); // Read return information
```

### SendDirectControlCommand Method

Sends a direct control command to a device. This is a member of the **Device** object.

*Note: The available direct control commands are determined by the individual device. See the device specific documentation for the device of interest. This command will only work...*
with Rapid Control Devices that have implemented the GetAutomationDeviceInfo method in their ConfigIF.

Returns
Long (4 byte integer value). The value returned is one of the following constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_OK (0)</td>
<td>Command was successfully submitted.</td>
</tr>
<tr>
<td>E_FAIL (a large negative number)</td>
<td>Failed to submit direct control command. This error code would also be returned if the instrument is opened offline.</td>
</tr>
</tbody>
</table>

Syntax
SendDirectControlCommand (Command)

Command Required Variant pointer. This is the command variant that will be sent to the device. The format of this command variant is determined by the individual device. See the device specific documentation for the device of interest.

Example
This example sends a direct control command to a device:

```
Dim vCommand as Variant
Dim resultErrorCode as Long

SetCommand(vCommand); // Sets information
resultErrorCode = SendDirectControlCommand(vCommand)
ProcessReturnCode(resultErrorCode)
```

VendorID Property

Returns the vendor (OEM) ID for this device. This is a member of the Device object.

Type
Read Only Long (4 byte integer value)

Example
This example displays a message box with the vendor ID for the device:

```
strMsg = "The vendor ID for the current " & "device is " & Device.VendorID
MsgBox (strMsg)
```
Projects Object

This object contains a collection of all the Project objects that are currently available. Applications can enumerate through the collection of projects by using For ... Next statements or by using For ... Each statements. For an overview of all objects, see the Object Hierarchy.

Example
This example displays the name of each project in the collection using For ... Each statements:

```vba
For Each Project In Projects
    MsgBox ("Project Name: " & Project.Name)
Next
```

This example displays the name of each project in the collection using For ... Next statements:

```vba
nCount = Projects.Count
For nIndex = 0 To nCount - 1
    MsgBox ("Project Name: " & Projects(nIndex).Name)
Next nIndex
```

Count Property

Returns the number of projects available. This is a member of the Projects object.

Type
Read Only Long (4 byte integer value)

Example
This example displays the name of each project to the user:

```vba
nCount = Projects.Count
For nIndex = 0 To nCount - 1
    MsgBox ("Project Name: " & Projects(nIndex).Name)
Next nIndex
```

Item Property

Returns the Project object at the given indexed location. This is a member of the Projects object.

Type
Read Only Object

Syntax
Item(Index)

**Index**
Required Long (4 byte integer value). This is a zero based index that specifies the Project object to be returned.

**Example**
This example displays the name of each project in the collection:

```
nCount = Projects.Count
For nIndex = 0 To nCount - 1
    MsgBox ("Project Name: " & Projects.Item(nIndex).Name)
Next nIndex
```

**Remarks**
Item is the default member of this collection, so the following two lines of code are equivalent:

```
Projects.Item(1)
Projects(1)
```

**Name Property**
Returns the name of this object. This is a member of the Projects object.

**Type**
Read Only String

**Example**
This example displays a message box with the name of this object to the user:

```
strMsg = "The name of the Projects object is " & _
Projects.Name
MsgBox (strMsg)
```
**Project Object**

This object represents a project that has been established by the system administrator. The **Projects** collection object contains all of the project objects that are currently available. For an overview of all objects, see the **Object Hierarchy**.

**DataPath Property**

Returns the data path of this project, as established by the system administrator. This is a member of the **Project** object.

**Type**

Read Only String

**Example**

This example displays the data path of this project to the user:

```vba
MsgBox ("Data Path: " & Project.DataPath)
```

**Description Property**

Returns the description of this project, as established by the system administrator. This is a member of the **Project** object.

**Type**

Read Only String

**Example**

This example displays the description of this project to the user:

```vba
MsgBox ("Project Description: " & Project.Description)
```

**MethodPath Property**

Returns the method path of this project, as established by the system administrator. This is a member of the **Project** object.

**Type**

Read Only String

**Example**

This example displays the method path of this project to the user:

```vba
MsgBox ("Method Path: " & Project.MethodPath)
```
**Name Property**

Returns the name of this project, as established by the system administrator. This is a member of the `Project` object.

**Type**
Read Only String

**Example**
This example displays the name of this project to the user:

```
MsgBox ("Project Name: " & Project.Name)
```

**ProjectPath Property**

Returns the project path of this project, as established by the system administrator. This is a member of the `Project` object.

**Type**
Read Only String

**Example**
This example displays the project path of this project to the user:

```
MsgBox ("Project Path: " & Project.ProjectPath)
```

**SequencePath Property**

Returns the sequence path of this project, as established by the system administrator. This is a member of the `Project` object.

**Type**
Read Only String

**Example**
This example displays the sequence path of this project to the user:

```
MsgBox ("Sequence Path: " & Project.SequencePath)
```

**TemplatePath Property**

Returns the report template path of this project, as established by the system administrator. This is a member of the `Project` object.

**Type**
Read Only String

Example
This example displays the report template path of this project to the user:

    MsgBox("Report Template Path: " & Project.TemplatePath)
Traces Object

This object contains a collection of all the Trace objects that are currently available. Traces are the actual data streams displayed in the graph of the data system. There are currently three types of traces available. These are Normal Chromatogram Traces, Auxiliary Traces, and 3D Traces. The raw data points may be retrieved for the Normal Chromatogram Traces, the Auxiliary Traces, and the 3D Traces. Applications can enumerate through the collection of traces by using For … Next statements or by using For … Each statements. For an overview of all objects, see the Object Hierarchy.

Example

This example displays the name of each trace in the collection using For … Each statements:

```vba
For Each Trace In Traces
    MsgBox ("Trace Name: " & Trace.Name)
Next
```

This example displays the name of each trace in the collection using For … Next statements:

```vba
nCount = Traces.Count
For nIndex = 0 To nCount - 1
    MsgBox ("Trace Name: " & Traces(nIndex).Name)
Next nIndex
```

Count Property

Returns the number of traces available. This is a member of the Traces object.

Type

Read Only Long (4 byte integer value)

Example

This example displays the name of each trace to the user:

```vba
nCount = Traces.Count
For nIndex = 0 To nCount - 1
    MsgBox ("Trace Name: " & _
            Traces(nIndex).Name)
Next nIndex
```

Item Property

Returns the Trace object at the given indexed location. This is a member of the Traces object.

Type

Read Only Object

Syntax
Item(Index)

Index  Required Long (4 byte integer value). This is a zero based index that specifies the Trace object to be returned.

Example
This example displays the name of each trace in the collection:

    nCount = Traces.Count
    For nIndex = 0 To nCount - 1
        MsgBox ("Trace Name: " & Traces.Item(nIndex).Name)
    Next nIndex

Remarks
Item is the default member of this collection, so the following two lines of code are equivalent:

    Traces.Item(1)
    Traces(1)

Name Property

Returns the name of this object. This is a member of the Traces object.

Type
Read Only String

Example
This example displays a message box with the name of this object to the user:

    strMsg = "The name of the Traces object is " & _
    Traces.Name
    MsgBox (strMsg)
**Trace Object**

This object represents a stream of data displayed in the graph of the data system. The Traces collection object contains all of the trace objects that are currently available. For an overview of all objects, see the Object Hierarchy.

**ActualPointCount Property**

Returns the actual number of data points that were collected for this trace. This is a member of the Trace object.

**Type**

Read Only Long (4 byte integer value)

**Example**

This example displays the actual number of points for this trace:

```vba
strMsg = "The actual point count is " & Trace.ActualPointCount
MsgBox (strMsg)
```

**ActualRunTime Property**

Returns the actual run time of this trace. This value is always returned in seconds. This is a member of the Trace object.

**Type**

Read Only Single (4 byte floating point value)

**Example**

This example displays the actual run time of this trace:

```vba
strMsg = "The actual run time is: " & Trace.ActualRunTime
MsgBox (strMsg)
```

**ASTMNoise Property**

Returns the ASTM Noise of this trace. This is a member of the Trace object.

Note: If the data file has not undergone a System Suitability analysis with an ASTM Noise test, a value of 0 will be returned.

**Type**

Read Only Double (8 byte floating point value)
Example
This example displays the ASTM Noise of this trace:

```vba
strMsg = "The ASTM Noise is: " & _
        Trace.ASTMNoise
MsgBox (strMsg)
```

**BaseFrequency Property**

Returns the base frequency of this trace. This is a member of the Trace object.

**Type**
Read Only Single (4 byte floating point value)

Example
This example displays the base frequency of this trace:

```vba
strMsg = "The base frequency of this trace is: " & _
        Trace.BaseFrequency
MsgBox (strMsg)
```

**DelayTime Property**

Returns the delay time of this trace. This value is always returned in seconds. This is a member of the Trace object.

**Type**
Read Only Single (4 byte floating point value)

Example
This example displays the delay time of this trace:

```vba
strMsg = "The delay time of this trace is: " & _
        Trace.DelayTime
MsgBox (strMsg)
```

**DetectorName Property**

Returns the name of the detector for this trace. This is a member of the Trace object.

**Type**
Read Only String

Example
This example displays the detector name of this trace to the user:

```vba
MsgBox ("Detector Name: " & Trace.DetectorName)
```
**Drift Property**

Returns the Drift of this trace. This is a member of the `Trace` object.

Note: If the data file has not undergone a System Suitability analysis with a Drift test, a value of 0 will be returned.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays the Drift of this trace:

```vba
strMsg = "The Drift is: " & _
    Trace.Drift
MsgBox (strMsg)
```

**ExpectedPointCount Property**

Returns the expected number of data points to be collected for this trace. This is a member of the `Trace` object.

**Type**
Read Only Long (4 byte integer value)

**Example**
This example displays the expected number of points for this trace:

```vba
strMsg = "The expected point count is " & _
    Trace.ExpectedPointCount
MsgBox (strMsg)
```

**ExpectedRunTime Property**

Returns the expected run time of this trace. This value is always returned in seconds. This is a member of the `Trace` object.

**Type**
Read Only Single (4 byte floating point value)

**Example**
This example displays the expected run time of this trace:

```vba
strMsg = "The expected run time is: " & _
    Trace.ExpectedRunTime
MsgBox (strMsg)```
GetTraceData Method

Returns the actual data points for this trace starting at the StartIndex and ending at the StartIndex plus the NumPtsToReturn. This is a member of the Trace object.

The value returned is one of the following constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Unknown error.</td>
</tr>
<tr>
<td>0</td>
<td>OK.</td>
</tr>
<tr>
<td>1</td>
<td>Unable to access the trace information.</td>
</tr>
<tr>
<td>2</td>
<td>There are no points available for this trace.</td>
</tr>
<tr>
<td>3</td>
<td>StartIndex is greater than the total points available.</td>
</tr>
<tr>
<td>4</td>
<td>Could not create a safe data array.</td>
</tr>
<tr>
<td>5</td>
<td>Unable to access the data.</td>
</tr>
<tr>
<td>6</td>
<td>Some other error occurred when getting the point data.</td>
</tr>
</tbody>
</table>

Returns
Long (4 byte integer value)

Syntax
GetTraceData(StartIndex, NumPtsToReturn, vXPoints, vYPoints)

StartIndex
Long (4 byte integer value). This is a required integer that indicates which point to start with.

NumPtsToReturn
Long (4 byte integer value). This is a required integer that indicates how many points to return. Set this value to 0 to get all points from the StartIndex on. Check the variant size to see how many points were actually returned.

vXPoints
Variant. This is a variant that, if the trace type is ezChromTrace or ez3DTrace, will not be used. If the trace type is ezAuxTrace, it will receive an array of Double (8 byte floating point values). Each value in this array represents a point of data that was collected for the X axis.

vYPoints
Variant. This is a variant that, if the trace type is ezChromTrace, will receive an array of Long (4 byte integer value) values, if the trace type is ezAuxTrace, will receive an array of Double (8 byte floating point values), and if the trace type is ez3DTrace, will receive an array of variant arrays of Long (4 byte floating point integer values). Each value in this array represents a point of data that was collected for the spectra.

Example
This example requests the specified data points for a trace, and then loops through each array to find the individual values:

```vbs
Dim vXData As Variant
Dim vYData As Variant

lResult = Trace.GetTraceData(10, 25, vXData, vYData)
If (lResult = 0) Then
```
If (VarType(vXData) <> vbEmpty) Then
    For Each dataXPoint In vXData
        XPoint = dataXPoint
    Next
End If

If (VarType(vYData) <> vbEmpty) Then
    For Each dataYPoint In vYData
        YPoint = dataYPoint
    Next
End If

GetTracePoints Method

Returns the actual data points for this trace. This is a member of the Trace object.

Returns
Void

Syntax
GetTracePoints(vXPoints, vYPoints)

vXPoints Variant. This is a variant that, if the trace type is ezChromTrace or ez3DTrace, will not be used. If the trace type is ezAuxTrace, it will receive an array of Double (8 byte floating point values). Each value in this array represents a point of data that was collected for the X axis.

vYPoints Variant. This is a variant that, if the trace type is ezChromTrace, will receive an array of Long (4 byte integer value) values, if the trace type is ezAuxTrace, will receive an array of Double (8 byte floating point values), and if the trace type is ez3DTrace, will receive an array of variant arrays of Long (4 byte floating point integer values). Each value in this array represents a point of data that was collected for the spectra.

Example
This example requests all of the data points for a trace, and then loops through each array to find the individual values:

Dim vXData As Variant
Dim vYData As Variant

Call Trace.GetTracePoints(vXData, vYData)
If (VarType(vXData) <> vbEmpty) Then
    For Each dataXPoint In vXData
        XPoint = dataXPoint
    Next
End If

If (VarType(vYData) <> vbEmpty) Then
    For Each dataYPoint In vYData
        YPoint = dataYPoint
    Next
End If
YPoint = dataYPoint
Next
End If

**Groups Property**

Returns the **Groups** object. This is a member of the **Trace** object.

**Type**
Read Only Object

**Example**
This example displays the name of each group to the user:

```
For Each Group In Trace.Groups
    MsgBox ("Group Name: " & Group.Name)
Next
```

**Name Property**

Returns the name of this trace. This is a member of the **Trace** object.

**Type**
Read Only String

**Example**
This example displays the name of this trace to the user:

```
MsgBox ("Trace Name: " & Trace.Name)
```

**Peaks Property**

Returns the **Peaks** object. This is a member of the **Trace** object.

**Type**
Read Only Object

**Example**
This example displays the name of each named peak to the user:

```
For Each Peak In Trace.Peaks
    bPeakNamed = Peak.Named
    if (bPeakNamed = True) Then
        MsgBox ("Peak Name: " & Peak.Name)
    End If
Next
```
**RefreshGroups Method**

Refreshes the Groups object, which reloads the Group list. This is a member of the Trace object.

**Returns**
Void

**Syntax**
RefreshGroups()

**Example**
This example displays the most recent list of Groups:

```vba
Trace.RefreshGroups
For Each Group In Trace.Groups
    MsgBox Group.Name
Next
```

---

**RefreshPeaks Method**

Refreshes the Peaks object, which reloads the Peak list. This is a member of the Trace object.

**Returns**
Void

**Syntax**
RefreshPeaks()

**Example**
This example displays the most recent list of Peaks:

```vba
Trace.RefreshPeaks
For Each Peak In Trace.Peaks
    MsgBox Peak.Name
Next
```

---

**RMSNoise Property**

Returns the RMS Noise of this trace. This is a member of the Trace object.

**Note:** If the data file has not undergone a System Suitability analysis with an RMS Noise test, a value of 0 will be returned.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays the RMS Noise of this trace:
strMsg = "The RMS Noise is: " & _
        Trace.RMSNoise
MsgBox (strMsg)

**SamplingPeriod Property**

Returns the sampling period used to acquire this trace. This is a member of the Trace object.

**Type**
Read Only Single (4 byte floating point value)

**Example**
This example displays the sampling period used to acquire this trace:

```vbnet
strMsg = "The sampling period is " & _
        Trace.SamplingPeriod
MsgBox (strMsg)
```

**SixSigmaNoise Property**

Returns the Six Sigma Noise of this trace. This is a member of the Trace object.

Note: If the data file has not undergone a System Suitability analysis with a 6 Sigma Noise test, a value of 0 will be returned.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays the Six Sigma Noise of this trace:

```vbnet
strMsg = "The Six Sigma Noise is: " & _
        Trace.SixSigmaNoise
MsgBox (strMsg)
```

**StartTime Property**

Returns the start time of this trace. This value is always returned in seconds. This is a member of the Trace object.

**Type**
Read Only Single (4 byte floating point value)

**Example**
This example displays the start time of this trace:

```vbnet
strMsg = "The start time of this trace is: " & _
        Trace.StartTime
MsgBox (strMsg)
```
**StopTime Property**

Returns the stop time of this trace. This value is always returned in seconds. This is a member of the Trace object.

**Type**

Read Only Single (4 byte floating point value)

**Example**

This example displays the stop time of this trace:

```vba
strMsg = "The stop time of this trace is: " & Bstr(Trace.StopTime)
MsgBox (strMsg)
```

**TotalChanArea Property**

Returns the total area of all peaks found in this trace. This is a member of the Trace object.

**Type**

Read Only Double (8 byte floating point value)

**Example**

This example displays the total channel area of this trace:

```vba
strMsg = "The area of the peaks in this channel is: " & Bstr(Trace.TotalChanArea)
MsgBox (strMsg)
```

**TotalChanHeight Property**

Returns the total height of all peaks found in this trace. This is a member of the Trace object.

**Type**

Read Only Double (8 byte floating point value)

**Example**

This example displays the total channel height of this trace:

```vba
strMsg = "The sum of the heights of the peaks in this channel is: " & Bstr(Trace.TotalChanHeight)
MsgBox (strMsg)
```

**Type Property**

Returns a flag to indicate the type of this trace. This is a member of the Trace object.

The value returned is one of the following EZTraceTypeFlags constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ezUndefinedTraceType
The trace is not a defined type.

ezChromTrace
The trace is a normal chromatogram trace.

ezAuxTrace
The trace is an auxiliary trace.

ez3DTrace
The trace is a 3D trace.

Type
Read Only Long (4 byte integer value)

Example
This example determines whether or not the current trace is an auxiliary trace, and displays a message box to the user:

If (Trace.Type = ezAuxTrace) Then
    MsgBox ("This is an auxiliary trace.")
Else
    MsgBox ("User is not an auxiliary trace.")
End If

UniformSampling Property
Returns TRUE or FALSE to indicate if the data points for this trace were uniformly sampled in the X direction. This is a member of the Trace object.

Type
Read Only Boolean

Example
This example displays a message box to indicate if the data points for this trace were uniformly sampled:

bUniformlySampled = Trace.UniformSampling
If (bUniformlySampled = True) Then
    MsgBox ("Data was uniformly sampled")
Else
    MsgBox ("Data was NOT uniformly sampled ")
End If

XAxisMultiplier Property
Returns the multiplier for the X-axis of this trace. This is a member of the Trace object.

Type
Read Only Single (4 byte floating point value)

Example
This example displays the multiplier used for the X-axis:

strMsg = "The X-axis multiplier is " & _
    Trace.XAxisMultiplier
MsgBox (strMsg)
**XAxisTitle Property**

Returns the X-axis title of this trace. This is a member of the Trace object.

**Type**  
Read Only String

**Example**  
This example displays the title of the X-axis to the user:

```
MsgBox ("X-Axis axis Title: " & Trace.XAxisTitle)
```

**NOTE:**  
The value returned here may not match what is seen on the graph if the value is a time. The graph checks the preferences for time values and will usually show either “Minutes” or “Seconds”.

---

**XAxisUnits Property**

Returns the X-axis units of this trace. This is a member of the Trace object.

**Type**  
Read Only String

**Example**  
This example displays the units of the X-axis to the user:

```
MsgBox ("X-Axis axis Units: " & Trace.XAxisUnits)
```

**NOTE:**  
The value returned here may not match what is seen on the graph if the value is a time. The graph checks the preferences for time values and will usually show either “Minutes” or “Seconds”. The XAxisUnits value will usually be the same as the XAxisTitle value.

---

**YAxisMultiplier Property**

Returns the multiplier for the Y-axis of this trace. This is a member of the Trace object.

**Type**  
Read Only Single (4 byte floating point value)

**Example**  
This example displays the multiplier used for the Y-axis:

```
strMsg = "The Y-axis multiplier is " & _
Trace.YAxisMultiplier
MsgBox (strMsg)
```
**YAxisTitle Property**

Returns the Y-axis title of this trace. This is a member of the Trace object.

**Type**

Read Only String

**Example**

This example displays the title of the Y-axis to the user:

```
_MsgBox_("Y-Axis axis Title: " & Trace.YAxisTitle)
```

**YAxisUnits Property**

Returns the Y-axis units of this trace. This is a member of the Trace object.

**Type**

Read Only String

**Example**

This example displays the units of the Y-axis to the user:

```
_MsgBox_("Y-Axis axis Units: " & Trace.YAxisUnits)
```
**FractionCollTraces Object**

This object contains a collection of all the `FractionCollTrace` objects that are currently available. All defined fraction collection traces will be available in the collection. The `Instrument` object returns a pointer to this object. Applications can enumerate through the collection of fraction collection traces by using `For ... Next` statements or by using `For ... Each` statements. For an overview of all objects, see the [Object Hierarchy](#).

**Example**

This example displays the fraction ID of each fraction collection trace in the collection using `For ... Each` statements:

```vbscript
For Each FractionCollTrace In FractionCollTraces
    MsgBox ("Fraction ID: " & FractionCollTrace.FractionID)
Next
```

This example displays the fraction ID of each fraction collection trace in the collection using `For ... Next` statements:

```vbscript
nCount = FractionCollTraces.Count
For nIndex = 0 To nCount - 1
    MsgBox ("Fraction ID: " &
             FractionCollTraces(nIndex).FractionID)
Next nIndex
```

**Count Property**

Returns the number of fraction collection traces available. This is a member of the `FractionCollTraces` object.

**Type**

Read Only Long (4 byte integer value)

**Example**

This example displays the fraction ID of each fraction collection trace to the user:

```vbscript
nCount = FractionCollTraces.Count
For nIndex = 0 To nCount - 1
    MsgBox ("Fraction ID: " &
             FractionCollTraces(nIndex).FractionID)
Next nIndex
```

**Item Property**

Returns the `FractionCollTrace` object at the given indexed location. This is a member of the `FractionCollTraces` object.
**Type**
Read Only Object

**Syntax**
**Item(Index)**

- **Index** Required Long (4 byte integer value). This is a zero based index that specifies the NamedPeak object to be returned.

**Example**
This example displays the fraction IDs of the fraction collection traces to the user:

```vba
nCount = FractionCollTraces.Count
For nIndex = 0 To nCount - 1
    MsgBox ("Fraction ID: " & _
             FractionCollTraces(nIndex).FractionID)
Next nIndex
```

**Remarks**
Item is the default member of this collection, so the following two lines of code are equivalent:

```vba
FractionCollTraces.Item(1)
FractionCollTraces(1)
```

---

**Name Property**
Returns the name of this object. This is a member of the **FractionCollTraces** object.

**Type**
Read Only String

**Example**
This example displays a message box with the name of this object to the user:

```vba
strMsg = "The name of the FractionCollTraces object is " & _
         FractionCollTraces.Name
MsgBox (strMsg)
```
**FractionCollTrace Object**

This object represents a fraction collection trace. The `FractionCollTraces` collection object contains all of the fraction collection trace objects that are currently available. For an overview of all objects, see the Object Hierarchy.

**FractionID Property**

Returns the fraction ID of this fraction collection trace. This is a member of the `FractionCollTrace` object.

**Type**

Read Only Long (4 byte integer value)

**Example**

This example displays a message box with the fraction ID of this fraction collection trace to the user:

```vbnet
strMsg = "The fraction ID of the current fraction " & "collection trace is " & FractionCollTrace.FractionID
MsgBox (strMsg)
```

**StartTime Property**

Returns the start time of this fraction collection trace. This is a member of the `FractionCollTrace` object.

**Type**

Read Only Double (8 byte floating point value)

**Example**

This example displays a message box with the start time of this fraction collection trace to the user:

```vbnet
strMsg = "The start time of the current fraction " & "collection trace is " & FractionCollTrace.StartTime
MsgBox (strMsg)
```

**StopTime Property**

Returns the stop time of this fraction collection trace. This is a member of the `FractionCollTrace` object.
**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the stop time of this fraction collection trace to the user:

```vbnet
strMsg = "The stop time of the current fraction " & _
         "collection trace is " & _
         FractionCollTrace.StopTime
MsgBox (strMsg)
```

**Vial_ID Property**
Returns the vial ID of this fraction collection trace. This is a member of the `FractionCollTrace` object.

**Type**
Read Only Long (4 byte integer value)

**Example**
This example displays a message box with the vial ID of this fraction collection trace to the user:

```vbnet
strMsg = "The vial ID of the current fraction " & _
         "collection trace is " & _
         FractionCollTrace.Vial_ID
MsgBox (strMsg)
```
Peaks Object

This object contains a collection of all the Peak objects that are currently available. All detected peaks and named peaks will be available in the collection. Applications can enumerate through the collection of peaks by using For … Next statements or by using For … Each statements. For an overview of all objects, see the Object Hierarchy.

Example
This example displays the name of each named peak in the collection using For … Each statements:

    For Each Peak In Peaks
        bNamedPeak = Peak.Named
        If (bNamedPeak = True) Then
            MsgBox ("Peak Name: " & Peak.Name)
        End If
    Next

This example displays the name of each named peak in the collection using For … Next statements:

    nCount = Peaks.Count
    For nIndex = 0 To nCount - 1
        bNamedPeak = Peaks(nIndex).Named
        If (bNamedPeak = True) Then
            MsgBox ("Peak Name: " & Peaks(nIndex).Name)
        End If
    Next nIndex

Count Property

Returns the number of peaks available. This count includes all detected and named peaks for the current trace. This is a member of the Peaks object.

Type
Read Only Long (4 byte integer value)

Example
This example displays the name of each named peak to the user:

    nCount = Peaks.Count
    For nIndex = 0 To nCount - 1
        bNamedPeak = Peaks(nIndex).Named
        If (bNamedPeak = True) Then
            MsgBox ("Peak Name: " & Peaks(nIndex).Name)
        End If
    Next nIndex
**Item Property**

Returns the *Peak* object at the given indexed location. This is a member of the *Peaks* object.

**Type**
Read Only Object

**Syntax**

```plaintext
Item(Index)
```

**Index**
Required Long (4 byte integer value). This is a zero based index that specifies the Peak object to be returned.

**Example**

This example displays the name of each named peak to the user:

```plaintext
nCount = Peaks.Count
For nIndex = 0 To nCount - 1
    bNamedPeak = Peaks.Item(nIndex).Named
    If (bNamedPeak = True) Then
        MsgBox("Peak Name: " & Peaks.Item(nIndex).Name)
    End If
Next nIndex
```

**Remarks**

Item is the default member of this collection, so the following two lines of code are equivalent:

```plaintext
Peaks.Item(1)
Peaks(1)
```

**Name Property**

Returns the name of this object. This is a member of the *Peaks* object.

**Type**

Read Only String

**Example**

This example displays a message box with the name of this object to the user:

```plaintext
strMsg = "The name of the Peaks object is " & _
        Peaks.Name
MsgBox (strMsg)
```
**Peak Object**

This object represents a named or detected peak found on a trace. The Peaks collection object contains all of the peak objects that are currently available. For an overview of all objects, see the Object Hierarchy.

**AmplitudeBaselineStart Property**

Returns the peak amplitude at the start of the baseline for this peak. This value is always returned in microvolts. This is a member of the Peak object.

**Note:** This property is only valid for detected peaks. See the Detected property.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the peak amplitude at the start of the baseline for the current peak:

```vbnet
strMsg = "The amplitude at the start of the baseline is " _
& Peak.AmplitudeBaselineStart
MsgBox (strMsg)
```

**AmplitudeBaselineStop Property**

Returns the peak amplitude at the end of the baseline for this peak. This value is always returned in microvolts. This is a member of the Peak object.

**Note:** This property is only valid for detected peaks. See the Detected property.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the peak amplitude at the end of the baseline for the current peak:

```vbnet
strMsg = "The amplitude at the end of the baseline is " _
& Peak.AmplitudeBaselineStop
MsgBox (strMsg)
```

**ApexTime Property**

Returns the apex time of this peak. This is the time at which the apex of the peak is detected. This value is always returned in minutes. This is a member of the Peak object.
Note: This property is only valid for detected peaks. See the Detected property.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the apex time of the current peak:

```vba
strMsg = "The apex time of the current peak is " & _
        Peak.ApexTime
MsgBox (strMsg)
```

**AreaPercent Property**
Returns the area percent of the current peak. This is a member of the Peak object.

Note: This property is only valid for detected peaks. See the Detected property.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the area percent for the current peak:

```vba
strMsg = "The area percent of the current " _
        & "peak is " & Peak.AreaPercent
MsgBox (strMsg)
```

**Asymmetry Property**
Returns the asymmetry of the current peak. This is a member of the Peak object.

Note: This property is only valid for detected peaks. See the Detected property.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the asymmetry of the current peak:

```vba
strMsg = "The asymmetry of the current peak is " _
        & Peak.Asymmetry
MsgBox (strMsg)
```

**AsymmetryTenPercent Property**
Returns the asymmetry of the current peak at 10% of its height. This is a member of the Peak object.
Note: This property is only valid for detected peaks. See the Detected property.

Type
Read Only Double (8 byte floating point value)

Example
This example displays a message box with the asymmetry of the current peak at 10% of the peak's height:

```vbscript
strMsg = "The asymmetry of the current peak at 10% " & "of its height is " & Peak.AsymmetryTenPercent
MsgBox (strMsg)
```

**BackInflectionIndex Property**
Returns an index into the data point array where the inflection point on the back of the peak occurs. This is a member of the Peak object.

Note: This property is only valid for detected peaks. See the Detected property.

Type
Read Only Double (8 byte floating point value)

Example
This example displays a message box with the index of the back inflection point of the current peak:

```vbscript
strMsg = "The index of the back inflection point is " & Peak.BackInflectionIndex
MsgBox (strMsg)
```

**BaselineStartTime Property**
Returns the start time of the baseline for this peak. This value is always returned in minutes. This is a member of the Peak object.

Note: This property is only valid for detected peaks. See the Detected property.

Type
Read Only Double (8 byte floating point value)

Example
This example displays a message box with the baseline start time of the current peak:

```vbscript
strMsg = "The start time of the baseline for the " & "current peak is " & Peak.BaselineStartTime
MsgBox (strMsg)
```
**BaselineStopTime Property**

Returns the stop time of the baseline for this peak. This value is always returned in minutes. This is a member of the Peak object.

*Note: This property is only valid for detected peaks. See the Detected property.*

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the baseline stop time of the current peak:

```vba
strMsg = "The stop time of the baseline for the " _
& "current peak is " & Peak.BaselineStopTime
MsgBox (strMsg)
```

**CapacityFactor Property**

Returns the capacity factor for the current peak. This is a member of the Peak object.

*Note: This property is only valid for detected peaks. See the Detected property.*

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the capacity factor for the current peak:

```vba
strMsg = "The capacity factor for the current " _
& "peak is " & Peak.CapacityFactor
MsgBox (strMsg)
```

**ConcentrationUnits Property**

Returns the concentration units of this peak. This is a member of the Peak object.

**Type**
Read Only String

**Example**
This example displays the concentration units of this peak to the user:

```vba
MsgBox ("Concentration Units: " & Peak.ConcentrationUnits)
```
**CorrectedArea Property**

Returns the corrected area of the current peak. This is a member of the Peak object.

**Note:** This property is only valid for detected peaks. See the Detected property.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the corrected area of the current peak:

```vba
strMsg = "The corrected area of the current peak is " _ & Peak.CorrectedArea
MsgBox (strMsg)
```

**Detected Property**

Returns TRUE or FALSE to indicate if the current peak was detected in the chromatogram. This is a member of the Peak object.

**Type**
Read Only Boolean

**Example**
This example displays a message box to indicate if the current peak was detected:

```vba
bDetectedPeak = Peak.Detected
If (bDetectedPeak = True) Then
    MsgBox ("Peak was detected.")
Else
    MsgBox ("Peak was NOT detected.")
End If
```

**DetectedPeakNumber Property**

Returns the detected peak number of this peak. If a valid peak number is returned, it will be a number greater than 0. This is a member of the Peak object.

**Note:** This property is only valid for detected peaks. See the Detected property.

**Type**
Read Only Long (4 byte integer value)

**Example**
This example displays the detected peak number of this peak to the user:

```vba
strMsg = "The detected peak number is " _ & Peak.DetectedPeakNumber
```
DownInflectionBaselineX Property

Returns the X intersect of the down inflection line with the baseline for this peak. This value is always returned in seconds. This is a member of the Peak object.

Note: This property is only valid for detected peaks. See the Detected property.

Type
Read Only Double (8 byte floating point value)

Example
This example displays a message box with the X intersect of the inflection line with the baseline:

```
strMsg = "The X intersect of the inflection line with " & "the baseline is " & Peak.DownInflectionBaselineX
MsgBox (strMsg)
```

DownInflectionBaselineY Property

Returns the Y intersect of the down inflection line with the baseline for this peak. This value is always returned in microvolts. This is a member of the Peak object.

Note: This property is only valid for detected peaks. See the Detected property.

Type
Read Only Double (8 byte floating point value)

Example
This example displays a message box with the Y intersect of the inflection line with the baseline:

```
strMsg = "The Y intersect of the inflection line with " & "the baseline is " & Peak.DownInflectionBaselineY
MsgBox (strMsg)
```

DownSlopeSimilarity Property

Returns the down slope similarity of the current peak. This is a member of the Peak object.

Note: This property is only valid for detected peaks. See the Detected property.

Type
Read Only Double (8 byte floating point value)

Example
This example displays a message box with the down slope similarity of the current peak:
strMsg = "The down slope similarity of this peak is " _
& Peak.DownSlopeSimilarity
MsgBox (strMsg)

**ESTDConcentration Property**

Returns the ESTD concentration of the current peak. This is a member of the Peak object.

**Note:** This property is only valid for detected peaks. See the Detected property.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the ESTD concentration for the current peak:

```vbnet
bConcentrationValid = Peak.ESTDConcentrationValid
If (bConcentrationValid = True) Then
    strMsg = "The ESTD concentration of this peak is " _
    & Peak.ESTDConcentration
    MsgBox (strMsg)
Else
    MsgBox ("ESTD concentration is not valid for this peak.")
End If
```

**ESTDConcentrationValid Property**

Returns TRUE or FALSE to indicate if the ESTD concentration is valid for the current peak. This is a member of the Peak object.

**Type**
Read Only Boolean

**Example**
This example displays a message box with the ESTD concentration for the current peak:

```vbnet
bConcentrationValid = Peak.ESTDConcentrationValid
If (bConcentrationValid = True) Then
    strMsg = "The ESTD concentration of this peak is " _
    & Peak.ESTDConcentration
    MsgBox (strMsg)
Else
    MsgBox ("ESTD concentration is not valid for this peak.")
End If
```
FrontInflectionIndex Property

Returns an index into the data point array where the inflection point on the front of the peak occurs. This is a member of the Peak object.

Note: This property is only valid for detected peaks. See the Detected property.

Type
Read Only Double (8 byte floating point value)

Example
This example displays a message box with the index of the front inflection point of the current peak:

```vba
strMsg = "The index of the front inflection point is " & _
        Peak.FrontInflectionIndex
MsgBox (strMsg)
```

GetCustomParameter Method

Returns the value of the specified peak custom parameter. This is a member of the Peak object.

Note: This property is only valid for detected peaks. See the Detected property.

Returns
Void

Syntax
GetCustomParameter(ParamID, vParamValue)

ParamID Long (4 byte integer value). This is the parameter id of the requested parameter. Custom parameters have a parameter id in the range of 2000 to 2500. The first custom parameter defined in a method will have a value of 2000. The next custom parameter will have a value of 2001, etc.

vParamValue Variant. This is a variant that will receive the value of the specified custom parameter. The type of the variant will be either Double or String, depending on the type of the custom parameter.

Example
This example gets the value of the peak custom parameter with a parameter id of 2005:

```vba
Call Peak.GetCustomParameter(2005, vParamValue)
strMsg = "The custom parameter value is " & vParamValue
MsgBox (strMsg)
```

GroupNumber Property

Returns the group number of this peak if it has been made part of a group. If a valid group number is returned, it will be a number greater than or equal to 0. The group number will be the last group
that the peak is assigned to, even though the peak may exist in other groups. This is a member of the Peak object.

Note: This property is only valid for detected peaks. See the Detected property.

**Type**
Read Only Long (4 byte integer value)

**Example**
This example displays the group number of the current peak if it has been made part of a group:

```vba
strMsg = "The group number of this peak is " _
        & Peak.GroupNumber
MsgBox (strMsg)
```

**Height Property**

Returns the height of the current peak at its apex. This value is always returned in microvolts. This is a member of the Peak object.

Note: This property is only valid for detected peaks. See the Detected property.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the height of the current peak:

```vba
strMsg = "The height of the current peak is " _
        & Peak.Height
MsgBox (strMsg)
```

**HeightPercent Property**

Returns the height percent of the current peak. This is a member of the Peak object.

Note: This property is only valid for detected peaks. See the Detected property.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the height percent for the current peak:

```vba
strMsg = "The height percent of the current " _
        & "peak is " & Peak.HeightPercent
MsgBox (strMsg)
```
**InflectionX Property**

Returns the X intersect of the up and down inflection lines for this peak. This value is always returned in seconds. This is a member of the Peak object.

*Note: This property is only valid for detected peaks. See the Detected property.*

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the X intersect of the up and down inflection lines:

```vba
strMsg = "The X intersect of the up and down inflection " _ & "lines is " & Peak. InflectionX
MsgBox (strMsg)
```

**InflectionY Property**

Returns the Y intersect of the up and down inflection lines for this peak. This value is always returned in microvolts. This is a member of the Peak object.

*Note: This property is only valid for detected peaks. See the Detected property.*

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the Y intersect of the up and down inflection lines:

```vba
strMsg = "The Y intersect of the up and down inflection " _ & "lines is " & Peak. InflectionY
MsgBox (strMsg)
```

**IntegrationCodes Property**

Returns the integration codes for this peak. This is a member of the Peak object.

*Note: This property is only valid for detected peaks. See the Detected property.*

**Type**
Read Only String

**Example**
This example displays the integration codes of this peak to the user:

```vba
MsgBox ("Integration Codes: " & Peak.IntegrationCodes)
```
**ISTDConcentration Property**

Returns the ISTD concentration of the current peak. This is a member of the Peak object.

*Note: This property is only valid for detected peaks. See the Detected property.*

**Type**

Read Only Double (8 byte floating point value)

**Example**

This example displays a message box with the ISTD concentration for the current peak:

```vba
bConcentrationValid = Peak.ISTDConcentrationValid
If (bConcentrationValid = True) Then
    strMsg = "The ISTD concentration of this peak is " & Peak.ISTDConcentration
    MsgBox (strMsg)
Else
    MsgBox ("ISTD concentration is not valid for this peak.")
End If
```

**ISTDConcentrationValid Property**

Returns TRUE or FALSE to indicate if the ISTD concentration is valid for the current peak. This is a member of the Peak object.

**Type**

Read Only Boolean

**Example**

This example displays a message box with the ISTD concentration for the current peak:

```vba
bConcentrationValid = Peak.ISTDConcentrationValid
If (bConcentrationValid = True) Then
    strMsg = "The ISTD concentration of this peak is " & Peak.ISTDConcentration
    MsgBox (strMsg)
Else
    MsgBox ("ISTD concentration is not valid for this peak.")
End If
```

**ISTDPeakNumber Property**

Returns the peak number of the ISTD peak for this peak. If a valid peak number is returned, it will be a number greater than 0. This is a member of the Peak object.

*Note: This property is only valid for detected peaks. See the Detected property.*
Type
Read Only Long (4 byte integer value)

Example
This example displays the peak number of the ISTD peak to the user:

```
strMsg = "The ISTD peak number is " _
& Peak.ISTDPeakNumber
MsgBox (strMsg)
```

LambdaMax Property
Returns the maximum absorbance at the apex of this peak. This is a member of the Peak object.

Note: This property is only valid for detected peaks. See the Detected property.

Type
Read Only Long (4 byte integer value)

Example
This example displays the lambda max of the current peak:

```
strMsg = "The lambda max of this peak is " _
& Peak.LambdaMax
MsgBox (strMsg)
```

Name Property
Returns the name of this peak. This is a member of the Peak object. If this peak is named, then the peak name will always be returned. If this peak is not named, but is part of a “Calibrated Range” or “Uncalibrated Range” group then the name of the group will be returned for the peak name.

Type
Read Only String

Example
This example displays the name of this peak to the user:

```
MsgBox ("Peak Name: " & Peak.Name)
```

Named Property
Returns TRUE or FALSE to indicate if the current peak is named. This is a member of the Peak object.

Type
Read Only Boolean

Example
This example displays a message box to indicate if the current peak is named:

```vbnet
bNamedPeak = Peak.Named
If (bNamedPeak = True) Then
    MsgBox ("Peak is named.")
Else
    MsgBox ("Peak is NOT named.")
End If
```

**NamedPeakNumber Property**

Returns the peak number of this peak if it has been named. If a valid peak number is returned, it will be a number greater than 0. This is a member of the Peak object.

**Type**

Read Only Long (4 byte integer value)

**Example**

This example displays the peak number of the current peak if it has been named:

```vbnet
bNamedPeak = Peak.Named
If (bNamedPeak = True) Then
    strMsg = "The peak number of this named peak is " 
    & Peak.NamedPeakNumber
    MsgBox (strMsg)
Else
    MsgBox ("This peak was not named. ")
End If
```

**NORMConcentration Property**

Returns the normal concentration of the current peak. This is a member of the Peak object.

**Note:** This property is only valid for detected peaks. See the Detected property.

**Type**

Read Only Double (8 byte floating point value)

**Example**

This example displays a message box with the normal concentration for the current peak:

```vbnet
bConcentrationValid = Peak.NORMConcentrationValid
If (bConcentrationValid = True) Then
    strMsg = "The normal concentration of this peak is " 
    & Peak.NORMConcentration
    MsgBox (strMsg)
Else
    MsgBox ("Normal concentration is not valid for " 
    & "this peak.")
End If
```
**NORMConcentrationValid Property**

Returns TRUE or FALSE to indicate if the normal concentration is valid for the current peak. This is a member of the Peak object.

**Type**
Read Only Boolean

**Example**
This example displays a message box with the normal concentration for the current peak:

```vbnet
bConcentrationValid = Peak.NORMConcentrationValid
If (bConcentrationValid = True) Then
    strMsg = "The normal concentration of this peak is " _
    & Peak.NORMConcentration
    MsgBox (strMsg)
Else
    MsgBox ("Normal concentration is not valid for " _
    & "this peak.")
End If
```

**Purity Property**

Returns the purity of the current peak. This is a member of the Peak object.

**Note:** This property is only valid for detected peaks. See the Detected property.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the purity of the current peak:

```vbnet
strMsg = "The purity of this peak is " _
    & Peak.Purity
MsgBox (strMsg)
```

**QuantitationMethod Property**

Returns a flag to indicate the method of quantitation used for this peak. This is a member of the Peak object.

The value returned is one of the following EZQuantitationFlags constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ezQuantitateArea</td>
<td>The peak is quantitated on area.</td>
</tr>
</tbody>
</table>
**ezQuantitateHeight**  
The peak is quantitated on height.

**ezQuantitateCounts**  
The peak is quantitated on counts.

Note: This property is only valid for detected peaks. See the Detected property.

**Type**  
Read Only Long (4 byte integer value)

**Example**  
This example displays the quantitation method used on the current peak:

```vba
If (Peak.QuantitationMethod = ezQuantitateArea) Then  
    MsgBox ("This peak is quantitated on area.")
ElseIf (Peak.QuantitationMethod = ezQuantitateHeight) Then  
    MsgBox ("This peak is quantitated on height.")
Else  
    MsgBox ("This peak is quantitated on counts.")
End If
```

**ReferencePeakNumber Property**  
Returns the peak number of the reference peak for this peak. If a valid peak number is returned, it will be a number greater than 0. This is a member of the Peak object.

**Type**  
Read Only Long (4 byte integer value)

**Example**  
This example displays the peak number of the reference peak to the user:

```vba
strMsg = "The reference peak number is "  
& Peak.ReferencePeakNumber
MsgBox (strMsg)
```

**RelativeRetentionTime Property**  
Returns the relative retention time of the current peak. This value is always returned in minutes. This is a member of the Peak object.

Note: This property is only valid for detected peaks. See the Detected property.

**Type**  
Read Only Double (8 byte floating point value)

**Example**  
This example displays a message box with the relative retention time of the current peak:

```vba
strMsg = "The relative retention time of the current 
```
Resolution Property

Returns the resolution of the current peak. If more than one performance calculation method is used, the resolution value will be calculated by the first performance calculation method that is enabled. This is a member of the Peak object.

Note: This property is only valid for detected peaks. See the Detected property.

Type
Read Only Double (8 byte floating point value)

Example
This example displays a message box with the resolution (first performance calculation method that is enabled) of the current peak:

```
strMsg = "The resolution of the current " _
       & "peak is " & Peak.Resolution
MsgBox (strMsg)
```

ResolutionUSP Property

Returns the resolution (USP) of the current peak. This is a member of the Peak object.

Note: This property is only valid for detected peaks. See the Detected property.

Type
Read Only Double (8 byte floating point value)

Example
This example displays a message box with the resolution (USP) of the current peak:

```
strMsg = "The resolution of the current " _
       & "peak is " & Peak.ResolutionUSP
MsgBox (strMsg)
```

ResolutionEMG Property

Returns the resolution (EMG) of the current peak. This is a member of the Peak object.

Note: This property is only valid for detected peaks. See the Detected property.

Type
Read Only Double (8 byte floating point value)

Example
This example displays a message box with the resolution (EMG) of the current peak:
strMsg = "The resolution of the current " _
   & "peak is " & Peak.ResolutionEMG
MsgBox (strMsg)

**ResolutionDAB Property**

Returns the resolution (DAB) of the current peak. This is a member of the Peak object.

*Note: This property is only valid for detected peaks. See the Detected property.*

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the resolution (DAB) of the current peak:

strMsg = "The resolution of the current " _
   & "peak is " & Peak.ResolutionDAB
MsgBox (strMsg)

**ResolutionAOH Property**

Returns the resolution (AOH) of the current peak. This is a member of the Peak object.

*Note: This property is only valid for detected peaks. See the Detected property.*

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the resolution (AOH) of the current peak:

strMsg = "The resolution of the current " _
   & "peak is " & Peak.ResolutionAOH
MsgBox (strMsg)

**ResolutionJP Property**

Returns the resolution (JP) of the current peak. This is a member of the Peak object.

*Note: This property is only valid for detected peaks. See the Detected property.*

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the resolution (JP) of the current peak:

strMsg = "The resolution of the current " _
   & "peak is " & Peak.ResolutionJP
ResponseFactor Property

Returns the response factor of the current peak. This is a member of the Peak object.

Note: This property is only valid for detected peaks. See the Detected property.

Type
Read Only Double (8 byte floating point value)

Example
This example displays a message box with the response factor for the current peak:

```
strMsg = "The response factor of the current " _
& "peak is " & Peak.ResponseFactor
MsgBox (strMsg)
```

SimilarityIndex Property

Returns the similarity index of the current peak. This is a member of the Peak object.

Note: This property is only valid for detected peaks. See the Detected property.

Type
Read Only Double (8 byte floating point value)

Example
This example displays a message box with the similarity index of the current peak:

```
strMsg = "The similarity index of this peak is " _
& Peak.SimilarityIndex
MsgBox (strMsg)
```

StandardPeakMultiplier Property

Returns the concentration multiplier for the standard peak of this peak. This is a member of the Peak object.

Type
Read Only Double (8 byte floating point value)

Example
This example displays a message box with the concentration multiplier of the standard peak:

```
strMsg = "The concentration multiplier of the standard " _
& "peak is " & Peak.StandardPeakMultiplier
MsgBox (strMsg)
```
**StandardPeakNumber Property**

Returns the peak number of the standard peak for this peak. If a valid peak number is returned, it will be a number greater than 0. This is a member of the Peak object.

**Type**
Read Only Long (4 byte integer value)

**Example**
This example displays the peak number of the standard peak to the user:

```vba
strMsg = "The standard peak number is " & Peak.StandardPeakNumber
MsgBox (strMsg)
```

**StartTime Property**

Returns the start time of this peak. This value is always returned in minutes. This is a member of the Peak object.

**Note:** This property is only valid for detected peaks. See the Detected property.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the start time of the current peak:

```vba
strMsg = "The start time of the current peak is " & _
         Peak.StartTime
MsgBox (strMsg)
```

**StopTime Property**

Returns the stop time of this peak. This value is always returned in minutes. This is a member of the Peak object.

**Note:** This property is only valid for detected peaks. See the Detected property.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the stop time of the current peak:

```vba
strMsg = "The stop time of the current peak is " & _
         Peak.StopTime
MsgBox (strMsg)
```
**TheoreticalPlates Property**

Returns the theoretical plates for the current peak. This is a member of the Peak object.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the theoretical plates for the current peak:

```vba
strMsg = "The theoretical plates for the current " _ & "peak is " & Peak.TheoreticalPlates
MsgBox (strMsg)
```

**TheoreticalPlatesPerMeter Property**

Returns the theoretical plates per meter for the current peak. This is a member of the Peak object.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the theoretical plates per meter for the current peak:

```vba
strMsg = "The theoretical plates per meter for the " _ & "current peak is " & Peak.TheoreticalPlatesPerMeter
MsgBox (strMsg)
```

**Type Property**

Returns a flag to indicate the type of this peak. This is a member of the Peak object.

The value returned is one of the following EZPeakTypeFlags constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ezUnknownPeakType</td>
<td>The type of the peak is not known.</td>
</tr>
<tr>
<td>ezTangentPeakType</td>
<td>The peak is a tangent peak.</td>
</tr>
<tr>
<td>ezSolventPeakType</td>
<td>The peak is a solvent peak.</td>
</tr>
</tbody>
</table>

**Note:** This property is only valid for detected peaks. See the Detected property.

**Type**
Read Only Long (4 byte integer value)

**Example**
This example displays the type of the current peak to the user:

```vbnet
If (Peak.Type = ezUnknownPeakType) Then
    MsgBox ("The type of this peak is not known.")
ElseIf (Peak.Type = ezTangentPeakType) Then
    MsgBox ("This is a tangent peak.")
Else
    MsgBox ("This is a solvent peak.")
End If
```

**UpInflectionBaselineX Property**

Returns the X intersect of the up inflection line with the baseline for this peak. This value is always returned in seconds. This is a member of the Peak object.

**Note:** This property is only valid for detected peaks. See the Detected property.

**Type**

Read Only Double (8 byte floating point value)

**Example**

This example displays a message box with the X intersect of the inflection line with the baseline:

```vbnet
strMsg = "The X intersect of the inflection line with " + 
        "the baseline is " & Peak.UpInflectionBaselineX
MsgBox (strMsg)
```

**UpInflectionBaselineY Property**

Returns the Y intersect of the up inflection line with the baseline for this peak. This value is always returned in microvolts. This is a member of the Peak object.

**Note:** This property is only valid for detected peaks. See the Detected property.

**Type**

Read Only Double (8 byte floating point value)

**Example**

This example displays a message box with the Y intersect of the inflection line with the baseline:

```vbnet
strMsg = "The Y intersect of the inflection line with " + 
        "the baseline is " & Peak.UpInflectionBaselineY
MsgBox (strMsg)
```

**UpSlopeSimilarity Property**

Returns the up slope similarity of the current peak. This is a member of the Peak object.
Note: This property is only valid for detected peaks. See the Detected property.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the up slope similarity of the current peak:

```vba
strMsg = "The up slope similarity of this peak is " & Peak.UpSlopeSimilarity
MsgBox (strMsg)
```

**USPWidth Property**
Returns the USP width of the current peak. This value is always returned in minutes. This is a member of the Peak object.

Note: This property is only valid for detected peaks. See the Detected property.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the USP width of the current peak:

```vba
strMsg = "The USP width of the current peak is " & Peak.USPWidth
MsgBox (strMsg)
```

**Width Property**
Returns the width of the current peak at its baseline. This value is always returned in minutes. This is a member of the Peak object.

Note: This property is only valid for detected peaks. See the Detected property.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the width of the current peak:

```vba
strMsg = "The width of the current peak is " & Peak.Width
MsgBox (strMsg)
```
**WidthAtFiftyPercent Property**

Returns the width of the current peak at 50% of its height. This value is always returned in minutes. This is a member of the `Peak` object.

Note: This property is only valid for detected peaks. See the `Detected` property.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the width at 50% value for the current peak:

```vbnet
strMsg = "The width at 50% value for the current peak is " _ & Peak.WidthAtFiftyPercent
MsgBox (strMsg)
```

**WidthAtFivePercent Property**

Returns the width of the current peak at 5% of its height. This value is always returned in minutes. This is a member of the `Peak` object.

Note: This property is only valid for detected peaks. See the `Detected` property.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the width at 5% value for the current peak:

```vbnet
strMsg = "The width at 5% value for the current peak is " _ & Peak.WidthAtFivePercent
MsgBox (strMsg)
```

**WidthAtTenPercent Property**

Returns the width of the current peak at 10% of its height. This value is always returned in minutes. This is a member of the `Peak` object.

Note: This property is only valid for detected peaks. See the `Detected` property.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the width at 10% value for the current peak:
strMsg = "The width at 10% value for the current peak is " 
& Peak.WidthAtTenPercent
MsgBox (strMsg)
Groups Object

This object contains a collection of all the Group objects that are currently available. All defined Calibrated Range and Named Peak groups will be available in the collection. Uncalibrated Range groups will not be included. Applications can enumerate through the collection of groups by using For … Next statements or by using For … Each statements. For an overview of all objects, see the Object Hierarchy.

Example

This example displays the name of each group in the collection using For … Each statements:

```vbnet
For Each Group In Groups
    MsgBox ("Group Name: " & Group.Name)
Next
```

This example displays the name of each group in the collection using For … Next statements:

```vbnet
nCount = Groups.Count
For nIndex = 0 To nCount - 1
    MsgBox ("Group Name: " & Groups(nIndex).Name)
Next nIndex
```

Count Property

Returns the number of groups available. This count includes all defined groups for the current trace. This is a member of the Groups object.

Type

Read Only Long (4 byte integer value)

Example

This example displays the name of each group to the user:

```vbnet
nCount = Groups.Count
For nIndex = 0 To nCount - 1
    MsgBox ("Group Name: " & Groups(nIndex).Name)
Next nIndex
```

Item Property

Returns the Group object at the given indexed location. This is a member of the Groups object.

Type

Read Only Object

Syntax
Item(Index)

Index Required Long (4 byte integer value). This is a zero based index that specifies the Group object to be returned.

Example
This example displays the name of each group to the user:

```vbnet
nCount = Groups.Count
For nIndex = 0 To nCount - 1
    MsgBox ("Group Name: " & Groups(nIndex).Name)
Next nIndex
```

Remarks
Item is the default member of this collection, so the following two lines of code are equivalent:

```vbnet
Groups.Item(1)
Groups(1)
```

Name Property

Returns the name of this object. This is a member of the Groups object.

Type
Read Only String

Example
This example displays a message box with the name of this object to the user:

```vbnet
strMsg = "The name of the Groups object is " & _
         Groups.Name
MsgBox (strMsg)
```
**Group Object**

This object represents a defined group for a trace. The Groups collection object contains all of the group objects that are currently available. For an overview of all objects, see the Object Hierarchy.

**AreaPercent Property**

Returns the area percent of the group. This is a member of the Group object.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the area percent for the group:

```vba
strMsg = "The area percent of the current " & "group is " & Group.AreaPercent
MsgBox (strMsg)
```

**ConcentrationUnits Property**

Returns the concentration units of this group. This is a member of the Group object.

**Type**
Read Only String

**Example**
This example displays the concentration units of this group to the user:

```vba
MsgBox ("Concentration Units: " & Group.ConcentrationUnits)
```

**CorrectedArea Property**

Returns the corrected area of the current group. This is a member of the Group object.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the corrected area of the current group:

```vba
strMsg = "The corrected area of the current group is " & & Group.CorrectedArea
MsgBox (strMsg)
```
**ESTDConcentration Property**

Returns the ESTD concentration of the current group. This is a member of the Group object.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the ESTD concentration for the current group:

```vba
bConcentrationValid = Group.ESTDConcentrationValid
If (bConcentrationValid = True) Then
  strMsg = "The ESTD concentration of this group is " _
  & Group.ESTDConcentration
  MsgBox (strMsg)
Else
  MsgBox ("ESTD concentration is not valid for this " _
    & "group.")
End If
```

**ESTDConcentrationValid Property**

Returns TRUE or FALSE to indicate if the ESTD concentration is valid for the current group. This is a member of the Group object.

**Type**
Read Only Boolean

**Example**
This example displays a message box with the ESTD concentration for the current group:

```vba
bConcentrationValid = Group.ESTDConcentrationValid
If (bConcentrationValid = True) Then
  strMsg = "The ESTD concentration of this group is " _
    & Group.ESTDConcentration
  MsgBox (strMsg)
Else
  MsgBox ("ESTD concentration is not valid for this " _
    & "group.")
End If
```

**GroupNumber Property**

Returns the group number of this group. If a valid group number is returned, it will be a number greater than or equal to 0. This is a member of the Group object.

**Type**
Read Only Long (4 byte integer value)
Example
This example displays the group number of the current group:

```
strMsg = "The group number of this group is " 
       & Group.GroupNumber
MsgBox (strMsg)
```

**Height Property**

Returns the height of the current group. This value is always returned in microvolts. This is a member of the Group object.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the height of the current group:

```
strMsg = "The height of the current group is " 
       & Group.Height
MsgBox (strMsg)
```

**HeightPercent Property**

Returns the height percent of the current group. This is a member of the Group object.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the height percent for the current group:

```
strMsg = "The height percent of the current " 
       & "group is " & Group.HeightPercent
MsgBox (strMsg)
```

**ISTDConcentration Property**

Returns the ISTD concentration of the current group. This is a member of the Group object.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the ISTD concentration for the current group:
bConcentrationValid = Group.ISTDConcentrationValid
If (bConcentrationValid = True) Then
    strMsg = "The ISTD concentration of this group is " _
        & Group.ISTDConcentration
    MsgBox (strMsg)
Else
    MsgBox ("ISTD concentration is not valid for this " _
        & "group.")
End If

**ISTDConcentrationValid Property**

Returns TRUE or FALSE to indicate if the ISTD concentration is valid for the current group. This is a member of the **Group** object.

**Type**
Read Only Boolean

**Example**
This example displays a message box with the ISTD concentration for the current group:

```
bConcentrationValid = Group.ISTDConcentrationValid
If (bConcentrationValid = True) Then
    strMsg = "The ISTD concentration of this group is " _
        & Group.ISTDConcentration
    MsgBox (strMsg)
Else
    MsgBox ("ISTD concentration is not valid for this " _
        & "group.")
End If
```

**ISTDPeakNumber Property**

Returns the peak number of the ISTD peak for this group. If a valid peak number is returned, it will be a number greater than 0. This is a member of the **Group** object.

**Type**
Read Only Long (4 byte integer value)

**Example**
This example displays the peak number of the ISTD peak to the user:

```
strMsg = "The ISTD peak number is " _
    & Group.ISTDPeakNumber
MsgBox (strMsg)
```
**Name Property**

Returns the name of this group. This is a member of the [Group](#) object.

**Type**
Read Only String

**Example**
This example displays the name of this group to the user:

```vba
MsgBox ("Group Name: " & Group.Name)
```

**NORMConcentration Property**

Returns the normal concentration of the current group. This is a member of the [Group](#) object.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the normal concentration for the current group:

```vba
bConcentrationValid = Group.NORMConcentrationValid
If (bConcentrationValid = True) Then
    strMsg = "The normal concentration of this group is " _
    & Group.NORMConcentration
    MsgBox (strMsg)
Else
    MsgBox ("Normal concentration is not valid for " _
        & "this group.")
End If
```

**NORMConcentrationValid Property**

Returns TRUE or FALSE to indicate if the normal concentration is valid for the current group. This is a member of the [Group](#) object.

**Type**
Read Only Boolean

**Example**
This example displays a message box with the normal concentration for the current group:

```vba
bConcentrationValid = Group.NORMConcentrationValid
If (bConcentrationValid = True) Then
    strMsg = "The normal concentration of this group is " _
    & Group.NORMConcentration
    MsgBox (strMsg)
Else
    MsgBox ("Normal concentration is not valid for " _
        & "this group.")
End If
```
& "this group.")
End If

**QuantitationMethod Property**

Returns a flag to indicate the method of quantitation used for this group. This is a member of the `Group` object.

The value returned is one of the following `EZQuantitationFlags` constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ezQuantitateArea</td>
<td>The group is quantitated on area.</td>
</tr>
<tr>
<td>ezQuantitateHeight</td>
<td>The group is quantitated on height.</td>
</tr>
<tr>
<td>ezQuantitateCounts</td>
<td>The group is quantitated on counts.</td>
</tr>
</tbody>
</table>

**Type**
Read Only Long (4 byte integer value)

**Example**
This example displays the quantitation method used on the current group:

```vbscript
If (Group.QuantitationMethod = ezQuantitateArea) Then
    MsgBox ("This group is quantitated on area.")
ElseIf (Group.QuantitationMethod = ezQuantitateHeight) Then
    MsgBox ("This group is quantitated on height.")
Else
    MsgBox ("This group is quantitated on counts.")
End If
```

**ReferencePeakNumber Property**

Returns the peak number of the reference peak for this group. If a valid peak number is returned, it will be a number greater than 0. This is a member of the `Group` object.

**Type**
Read Only Long (4 byte integer value)

**Example**
This example displays the peak number of the reference peak to the user:

```vbscript
strMsg = "The reference peak number is " _
         & Group.ReferencePeakNumber
MsgBox (strMsg)
```
ResponseFactor Property
Returns the response factor of the current group. This is a member of the Group object.

Type
Read Only Double (8 byte floating point value)

Example
This example displays a message box with the response factor for the current group:

    strMsg = "The response factor of the current " _
      & "group is " & Group.ResponseFactor
    MsgBox (strMsg)

StandardPeakMultiplier Property
Returns the concentration multiplier for the standard peak of this group. This is a member of the Group object.

Type
Read Only Double (8 byte floating point value)

Example
This example displays a message box with the concentration multiplier of the standard peak:

    strMsg = "The concentration multiplier of the standard " _
      & "peak is " & Group.StandardPeakMultiplier
    MsgBox (strMsg)

StandardPeakNumber Property
Returns the peak number of the standard peak for this group. If a valid peak number is returned, it will be a number greater than 0. This is a member of the Group object.

Type
Read Only Long (4 byte integer value)

Example
This example displays the peak number of the standard peak to the user:

    strMsg = "The standard peak number is " _
      & Group.StandardPeakNumber
    MsgBox (strMsg)
MethodFileInfo Object

This object contains the methods and properties needed to obtain information about the current method file. The Instrument object returns a pointer to this object. For an overview of all objects, see the Object Hierarchy.

AddNewCustomParam Method

Adds a new custom parameter to the custom parameters list. This is a member of the MethodFileInfo object.

Returns
Boolean

Syntax
AddNewCustomParam(ParamName)

ParamName Required String. This is the name of the new custom parameter.

Example
This example adds a new custom parameter:

Dim bCreatedCustParamOK As Boolean
bCreatedCustParamOK = MethodFileInfo.AddNewCustomParam("abc")

NOTE: Use the MethodCustomParam interface to set the other new custom parameter properties. Then call the SetMethodCustomParamData method.

AddPDA_MultiChrom Method

Adds a new multi-chromatogram to the PDA multi-chromatogram table. This is a member of the MethodFileInfo object.

Returns
Boolean

If the data is valid and the table was not full, the return value should be True.

Syntax
AddPDA_MultiChrom(Enabled, Wavelength, Bandwidth)

Enabled Required Boolean. This indicates whether the multi-chromatogram will be enabled or not.

Wavelength Required Long (4 byte integer value). This is the multi-chromatogram wavelength.

Bandwidth Required Long (4 byte integer value). This is the multi-chromatogram bandwidth.

Example
This example adds a new multi-chromatogram to the PDA multi-chromatogram table:
Dim bAddedNewMultiChromOK As Boolean
bAddedNewMultiChromOK = _
    MethodFileInfo.AddPDA_MultiChrom(True, 254, 8)

**AfterAnalysisAdditionalParams Property**
Sets or returns the After Analysis additional parameters to be used for the After Analysis program. This is a member of the MethodFileInfo object.

**Type**
Read/Write String

**Example**
This example sets the After Analysis additional parameters:

    MethodFileInfo.AfterAnalysisAdditionalParams = "/p1 /p2"

**AfterAnalysisFileName Property**
Sets or returns the After Analysis program name. This is a member of the MethodFileInfo object.

**Type**
Read/Write String

**Example**
This example sets the After Analysis program name:

    MethodFileInfo.AfterAnalysisFileName = _
        "C:\EZChrom Elite\AfterAnalysis.exe"

**AfterExportAdditionalParams Property**
Sets or returns the After Export additional parameters to be used for the After Export program. This is a member of the MethodFileInfo object.

**Type**
Read/Write String

**Example**
This example sets the After Export additional parameters:

    MethodFileInfo.AfterExportAdditionalParams = "/p1 /p2"
**AfterExportFileName Property**

Sets or returns the After Export program name. This is a member of the `MethodInfo` object.

**Type**
Read/Write String

**Example**
This example sets the After Export program name:

```vbnet
MethodInfo.AfterExportFileName = _
"C:\EZChrom Elite\AfterExport.exe"
```

**BaselineFileName Property**

Sets or returns the Baseline data file name. This is a member of the `MethodInfo` object.

**Type**
Read/Write String

**Example**
This example sets the Baseline data file name:

```vbnet
MethodInfo.BaselineFileName = _
"C:\EZChrom Elite\Baseline.dat"
```

**BeforeAnalysisAdditionalParams Property**

Sets or returns the Before Analysis additional parameters to be used for the Before Analysis program. This is a member of the `MethodInfo` object.

**Type**
Read/Write String

**Example**
This example sets the Before Analysis additional parameters:

```vbnet
MethodInfo.BeforeAnalysisAdditionalParams = "/p1 /p2"
```

**BeforeAnalysisFileName Property**

Sets or returns the Before Analysis program name. This is a member of the `MethodInfo` object.

**Type**
Read/Write String

**Example**
This example sets the Before Analysis program name:
MethodFileInfo.BeforeAnalysisFileName = "C:\EZChrom Elite\BeforeAnalysis.exe"

**BeforeRunAdditionalParams Property**

Sets or returns the Before Run additional parameters to be used for the Before Run program. This is a member of the MethodFileInfo object.

**Type**
Read/Write String

**Example**
This example sets the Before Run additional parameters:

```
MethodFileInfo.BeforeRunAdditionalParams = "/p1 /p2"
```

**BeforeRunFileName Property**

Sets or returns the Before Run program name. This is a member of the MethodFileInfo object.

**Type**
Read/Write String

**Example**
This example sets the Before Run program name:

```
MethodFileInfo.BeforeRunFileName = "C:\EZChrom Elite\BeforeRun.exe"
```

**ChangePDA_MultiChrom Method**

Change the values for a particular multi-chromatogram in the PDA multi-chromatogram table.

**Returns**
Boolean

If the row number exists and the data is valid, this return value should return True.

**Syntax**
ChangePDA_MultiChrom(RowNumber, Enabled, Wavelength, Bandwidth)

- **RowNumber** Required Long (4 byte integer value). This is the multi-chromatogram row number in the multi-chromatogram table. Row numbers start at 1.
- **Enabled** Required Boolean. This indicates whether the multi-chromatogram will be enabled or not.
**Wavelength** Required Long (4 byte integer value). This is the multi-chromatogram wavelength.

**Bandwidth** Required Long (4 byte integer value). This is the multi-chromatogram bandwidth.

**Example**
This example changes the third multi-chromatogram in a table that has at least three multi-chromatograms:

```vba
Dim bChangeMultiChromOK As Boolean
bChangeMultiChromOK = MethodFileInfo.ChangePDA_MultiChrom(3, True, 254, 8)
```

**ClearAllMethodProgramSteps Method**

Clears all method program tables.

**Returns**
Long (4 byte integer value)

0 – Success: Programs cleared.

**Syntax**
ClearAllMethodProgramSteps()

**Example**
This example clears all method program tables:

```vba
Dim lRet as Long
lRet = MethodFileInfo.ClearAllMethodProgramSteps()
```

**DeleteCustomParam Method**

Deletes a custom parameter from the custom parameters list. This is a member of the MethodFileInfo object.

**Returns**
Boolean

**Syntax**
DeleteCustomParam(ParamName)

**ParamName** Required String. This is the name of the custom parameter to be deleted.

**Example**
This example deletes a custom parameter:
Dim bDelCustParamOK As Boolean
bDelCustParamOK = MethodFileInfo.DeleteCustomParam("abc")

**ExportEnabled Property**

Enables and disables export for the current method file. This is a member of the MethodFileInfo object.

**Type**
Read/Write Boolean

**Example**
This example enables export.

    MethodFileInfo.ExportEnabled = TRUE

This example disables export.

    MethodFileInfo.ExportEnabled = FALSE

This example gets the value of ExportEnabled.

    Dim bExportEnabled As Boolean
    bExportEnabled = MethodFileInfo.ExportEnabled

**GetAuditTrailMode Method**

Gets the audit trail mode for the current method file. This is a member of the MethodFileInfo object.

**Returns**
Long (4 byte integer value)

The return value will be one of the following:

0 – Prompt for reason at every change
1 – Prompt for reason when saving method
2 – Do not prompt for reason
any other value – undefined, audit trail may not be enabled

**Syntax**

GetAuditTrailMode ()

**Example**
This example gets the audit trail mode after checking that the audit trail is enabled:

    Dim bAuditTrailEnabled as Boolean
    Dim lMode as Long
    
    bAuditTrailEnabled = MethodFileInfo.IsAuditTrailEnabled
    If (bAuditTrailEnabled) then
        lMode = MethodFileInfo.GetAuditTrailMode()
    Else
' audit trail mode is undefined
End If

**GetChannelIDsToSubtract Method**

Returns the Channel IDs to subtract from a baseline program. This is a member of the [MethodFileInfo](#) object.

**Returns**

Void

**Syntax**

`GetChannelIDsToSubtract vChannelIDValues`

- **vChannelIDValues**: Required Variant. This is a variant that will receive an array of String values. Each value in this array represents a channel ID to be subtracted. If no channel IDs are to be subtracted, then this variant type will be set to vbEmpty.

**Example**

This example requests the list of channel IDs to subtract from a baseline program, and then loops through the array to find the individual values:

```vbscript
Dim vChannelIds As Variant
Dim channelID as String
Dim chanID as String

MethodFileInfo.GetChannelIDsToSubtract vChannelIds
If (VarType(vChannelIds) <> vbEmpty) Then
    For Each channelID In vChannelIds
        chanID = channelID
    Next
End If
```

**GetMethodFirstProgramValue Method**

Gets instrument specific item values from the first program step. If a program table contains initial values, the first step would be the step after the initial values. Initial values can be obtained using the GetMethodItemValue method. This is a member of the [MethodFileInfo](#) object.

**Returns**

Long (4 byte integer value)

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success: Value returned Successfully.</td>
</tr>
<tr>
<td>101</td>
<td>Failed: Item not supported by device.</td>
</tr>
<tr>
<td>103</td>
<td>Failed: No program steps.</td>
</tr>
</tbody>
</table>

**Syntax**

`GetMethodFirstProgramValue (ItemID, ByRef Time, ByRef ItemValue)`
**ItemID**
Required Long (4 byte integer value). The ID of the method item to get.
For a list of ItemIDs, refer to [Method Item IDs](#).
Run Time, ItemID 1000, is not supported by this method.

**Time**
Returns Long (4 byte integer value). The program time in seconds.

**ItemValue**
Returns Double (8 byte floating point value). The value of the item.

**Example**
This example gets pump flow program values:

```
Dim Value as Double
Dim Time as Long
Dim lRet as Long
lRet = MethodFileInfo.GetMethodFirstProgramValue(1010, Time, Value)
Do While ((lRet <> 103) And (lRet <> 101))
  lRet = MethodFileInfo.GetMethodNextProgramValue(1010, Time, Value)
Loop
```

**GetMethodItemValue Method**
Gets instrument specific method item values from the current open method. This method should also be used to get initial values for program tables even if the values appear to be part of the table. This is a member of the [MethodFileInfo](#) object.

**Returns**
Long (4 byte integer value)

- 0 – Success: Value returned Successfully.
- 101 – Failed: Item not supported by device.

**Syntax**
```
GetMethodItemValue (ItemID, ByRef ItemValue)
```

**ItemID**
Required Long (4 byte integer value). The ID of the method item to get.
For a list of ItemIDs, refer to [Method Item IDs](#).

**ItemValue**
Returns Double (8 byte floating point value). The value of the item.

**Example**
This example gets the initial value for Wavelength 1:

```
Dim dValue as Double
Dim lRet as Long
lRet = MethodFileInfo.GetMethodItemValue(1001, dValue)
```

**GetMethodNextProgramValue Method**
Gets instrument specific item values from the next program step. This should be called after the GetMethodFirstProgramValue method. This is a member of the [MethodFileInfo](#) object.

**Returns**
Long (4 byte integer value)
0 – Success: Value returned Successfully.
101 – Failed: Item not supported by device.
103 – Failed: Reached last program step.

**Syntax**

**GetMethodNextProgramValue** (ItemID, ByRef Time, ByRef ItemValue)

- **ItemID**
  - Required Long (4 byte integer value). The ID of the method item to get. For a list of ItemIDs, refer to Method Item IDs. Run Time, ItemID 1000, is not supported by this method.

- **Time**
  - Returns Long (4 byte integer value). The program time in seconds.

- **ItemValue**
  - Returns Double (8 byte floating point value). The value of the item.

**Example**

This example gets % Solvent B program values:

```vbnet
Dim Value as Double
Dim Time as Long
Dim lRet as Long
lRet = MethodFileInfo.GetMethodFirstProgramValue(1011, Time, Value)
Do While (lRet = 0)
  lRet = MethodFileInfo.GetMethodNextProgramValue(1011, Time, Value)
Loop
```

---

**GetPDA_BOOLData Method**

Gets boolean type PDA values. This is a member of the MethodFileInfo object.

**Returns**

Boolean

**Syntax**

**GetPDA_BOOLData** (ItemID, ByRef ItemValue)

- **ItemID**
  - Required Long (4 byte integer value). The ID of the PDA item to get.

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>104121</td>
<td>Spectrum Background Correction Flag</td>
</tr>
<tr>
<td>104122</td>
<td>Spectrum Interpolate Spectra Flag</td>
</tr>
<tr>
<td>104202</td>
<td>Spectrum Similarity Calculation Type Flag</td>
</tr>
<tr>
<td>104203</td>
<td>Spectrum Up Slope Calculation Type Flag</td>
</tr>
<tr>
<td>104204</td>
<td>Spectrum Down Slope Calculation Type Flag</td>
</tr>
<tr>
<td>104205</td>
<td>Spectrum Lambda Max Calculation Type Flag</td>
</tr>
</tbody>
</table>

- **ItemValue**
  - Returns Boolean. The value of the item.

**Example**

This example gets the Spectrum Background Correction Flag value, a PDA boolean type value:

```vbnet
Dim Value as Boolean
Dim bGetPDABoolValueOK As Boolean
bGetPDABoolValueOK = _
```
GetPDA_INT32Data Method

Gets long type PDA values. This is a member of the MethodFileInfo object.

Returns
Boolean

Syntax
GetPDA_INT32Data (ItemID, ByRef ItemValue)

<table>
<thead>
<tr>
<th>ItemID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>104100</td>
<td>Ratio Bandwidth 1</td>
</tr>
<tr>
<td>104101</td>
<td>Ratio Bandwidth 2</td>
</tr>
<tr>
<td>104102</td>
<td>Ratio Wavelength 1</td>
</tr>
<tr>
<td>104103</td>
<td>Ratio Wavelength 2</td>
</tr>
<tr>
<td>104104</td>
<td>Ratio Threshold</td>
</tr>
<tr>
<td>104120</td>
<td>Spectrum Filter Type</td>
</tr>
<tr>
<td>104125</td>
<td>Spectrum Peak Type</td>
</tr>
<tr>
<td>104126</td>
<td>Spectrum Average N Spectra</td>
</tr>
<tr>
<td>104127</td>
<td>Spectrum Lambda Max From</td>
</tr>
<tr>
<td>104128</td>
<td>Spectrum Lambda Max To</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ItemValue</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Required Long (4 byte integer value). The ID of the PDA item to get.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Returns Long (4 byte integer value). The value of the item.</td>
</tr>
<tr>
<td></td>
<td>For the Spectrum Filter Type, the value returned will be one of the following:</td>
</tr>
<tr>
<td></td>
<td>ID</td>
</tr>
<tr>
<td></td>
<td>200000</td>
</tr>
<tr>
<td></td>
<td>200001</td>
</tr>
<tr>
<td></td>
<td>200002</td>
</tr>
<tr>
<td></td>
<td>200003</td>
</tr>
<tr>
<td></td>
<td>For the Spectrum Peak Type, the value returned will be one of the following:</td>
</tr>
<tr>
<td></td>
<td>ID</td>
</tr>
<tr>
<td></td>
<td>104140</td>
</tr>
<tr>
<td></td>
<td>104141</td>
</tr>
<tr>
<td></td>
<td>104142</td>
</tr>
</tbody>
</table>

Example
This example gets the Spectrum Peak Type value, a PDA long type value:

Dim Value as Long
Dim bGetPDALongValueOK As Boolean
bGetPDALongValueOK = _
    MethodFileInfo.GetPDA_INT32Data(104125, Value)
GetPDA_MultiChrom Method

Gets PDA multi-chromatogram values for a specified row in the multi-chromatogram table. This is a member of the MethodFileInfo object.

**Returns**

Boolean

If the row number does not exist, the return value will be FALSE.

**Syntax**

GetPDA_MultiChrom(RowNumber, Enabled, Wavelength, Bandwidth)

- **RowNumber** Required Long (4 byte integer value). This is the multi-chromatogram row number in the multi-chromatogram table. Row numbers start at 1.
- **Enabled** Returns Boolean. This indicates whether the multi-chromatogram is enabled or not.
- **Wavelength** Returns Long (4 byte integer value). This is the multi-chromatogram wavelength.
- **Bandwidth** Returns Long (4 byte integer value). This is the multi-chromatogram bandwidth.

**Example**

This example gets PDA multi-chromatogram values for the first row of the multi-chromatogram table:

```vbnet
dim bEnabled as Boolean
dim lWavelength as Long
dim lBandwidth as Long
dim bGetMultiChromOK As Boolean
bGetMultiChromOK = MethodFileInfo.GetPDA_MultiChrom(1, _
    bEnabled, lWavelength, lBandwidth)
```

GetPDA_MultiChromTableSize Method

Gets the number of multi-chromatograms in the PDA multi-chromatogram table.

**Returns**

Long (4 byte integer value)

**Syntax**

GetPDA_MultiChromTableSize()

**Example**

This example gets the number of PDA multi-chromatograms:

```vbnet
dim lTotalPDAMultiChroms as Long
lTotalPDAMultiChroms = MethodFileInfo.GetPDA_MultiChromTableSize()
```
GetValidValueRange Method

Gets the min and max values for the specified item. This is a member of the MethodFileInfo object.

Returns
Long (4 byte integer value)

0 – Success: Values returned Successfully.
101 – Failed: Item not supported by device.

Syntax
GetValidValueRange (ItemID, ByRef MinItemValue, ByRef MaxItemValue)

ItemID Required Long (4 byte integer value). The ID of the method item to get. For a list of ItemIDs, refer to Method Item IDs.

MinItemValue Returns Double (8 byte floating point value). The minimum allowable value for the item.

MaxItemValue Returns Double (8 byte floating point value). The maximum allowable value for the item.

Example
This example gets the value range for oven temperature:

Dim MinValue as Double
Dim MaxValue as Double
Dim lRet as Long
lRet = MethodFileInfo.GetValidValueRange (1020, MinValue, MaxValue)

NOTE: For certain instruments 0 is a valid value and outside the range MinItemValue to MaxItemValue. In this case GetValidValueRange will only return MinItemValue and MaxItemValue. Other methods need to be used to determine if 0 is also valid value.

IsAuditTrailEnabled Method

Returns the audit trail enabled state for the open method. This is a member of the MethodFileInfo object.

Returns
Boolean

Syntax
IsAuditTrailEnabled()
**Method Item IDs**

The following are the IDs used by certain `MethodFileInfo` methods to get and set method properties.

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>Run Time –</td>
<td>For detectors this is the acquisition run time. For other devices this is the run time of the device.</td>
</tr>
<tr>
<td>1001</td>
<td>Wavelength 1</td>
<td>Detector’s first wavelength. For single channel detectors, this is the detector’s wavelength.</td>
</tr>
<tr>
<td>1002</td>
<td>Wavelength 2</td>
<td>Detector’s second wavelength. This is only supported by multiple channel detectors.</td>
</tr>
<tr>
<td>1003</td>
<td>Wavelength 3</td>
<td>Detector’s third wavelength. This is only supported by multiple channel detectors.</td>
</tr>
<tr>
<td>1004</td>
<td>Wavelength 4</td>
<td>Detector’s fourth wavelength. This is only supported by multiple channel detectors.</td>
</tr>
<tr>
<td>1005</td>
<td>Wavelength 5</td>
<td>Detector’s fifth wavelength. This is only supported by multiple channel detectors.</td>
</tr>
<tr>
<td>1010</td>
<td>Pump Flow –</td>
<td>Pump's total flow rate</td>
</tr>
<tr>
<td>1011</td>
<td>% Solvent B</td>
<td>Percentage contribution of pump Solvent B. When setting this value changes the existing value of Solvent B, Solvent A is automatically adjusted by the same amount to compensate. If Solvent A reaches 0% or 100%, Solvent C and then, if needed, Solvent D, are adjusted as needed to keep the overall solvent composition at 100%. For reproducible results set Solvent B, then C, then D.</td>
</tr>
<tr>
<td>1012</td>
<td>% Solvent C</td>
<td>Percentage contribution of pump Solvent C. When setting this value changes the existing value of Solvent C, Solvent A is automatically adjusted by the same amount to compensate. If Solvent A reaches 0% or 100%, Solvent D and then, if needed, Solvent B, are adjusted as needed to keep the overall solvent composition at 100%. For reproducible results set Solvent B, then C, then D.</td>
</tr>
<tr>
<td>1013</td>
<td>% Solvent D</td>
<td>Percentage contribution of pump Solvent D. When setting this value changes the existing value of Solvent D, Solvent A is automatically adjusted by the same amount to compensate. If Solvent A reaches 0% or 100%, Solvent B and then, if needed, Solvent C, are adjusted as needed to keep the overall solvent composition at 100%. For reproducible results set Solvent B, then C, then D.</td>
</tr>
<tr>
<td>1014</td>
<td>Gradient Curve</td>
<td>Pump curve parameter, which defines the gradient ramp profile.</td>
</tr>
<tr>
<td>1020</td>
<td>Oven Temperature</td>
<td>Column oven temperature</td>
</tr>
<tr>
<td>1025</td>
<td>Valve Position</td>
<td>Column or solvent switching valve position</td>
</tr>
</tbody>
</table>

Depending on the `Instrument` object's configuration and `MethodFileInfo` method, some Method Item IDs will not be supported. The user will need to verify which Method Item IDs are supported on the instrument to which they are connected.

**Example**
In this example the `GetMethodItemValue` method uses the Method Item ID of 1001 to get the initial value of Wavelength 1:

```vba
Dim dValue as Double
Dim lRet as Long
lRet = MethodFileInfo.GetMethodItemValue(1001, dValue)
```

**MethodCustomParams Property**

Returns the `MethodCustomParams` object. This is a member of the `MethodFileInfo` object.

**Type**
Read Only Object

**Example**
This example displays the name of each custom parameter to the user:

```vba
For Each MethodCustomParam In MethodFileInfo.MethodCustomParams
    MsgBox ("Method Custom Parameter Name: " & MethodCustomParam.ParameterName)
Next
```

**MethodPropAnalysisPeriod Property**

Sets or returns the method properties option Analysis Period. This is a member of the `MethodFileInfo` object.

**Type**
Read/Write Double (8 byte floating point value)

The units for this value is seconds.

**Example**
This example sets the analysis period:

```vba
MethodFileInfo.MethodPropAnalysisPeriod = 7.2
```

**MethodPropAnalyzeAfterAcq Property**

Sets or returns the method properties option Analyze After Acquisition. This is a member of the `MethodFileInfo` object.

**Type**
Read/Write Boolean

**Example**
This example sets the analyze after acquisition option:

```vba
MethodFileInfo.MethodPropAnalyzeAfterAcq = True
```
**MethodPropAnalyzeDuringAcq Property**
Sets or returns the method properties option Analyze During Acquisition. This is a member of the `MethodFileInfo` object.

**Type**
Read/Write Boolean

**Example**
This example sets the analyze during acquisition option:

```plaintext
MethodFileInfo.MethodPropAnalyzeDuringAcq = True
```

**MethodPropAutoAvgReplicates Property**
Sets or returns the method properties option Auto Average Replicates. This is a member of the `MethodFileInfo` object.

**Type**
Read/Write Boolean

**Example**
This example sets the auto average replicates option:

```plaintext
MethodFileInfo.MethodPropAutoAvgReplicates = True
```

**MethodPropEnableDataFileCompress Property**
Sets or returns the method properties option Enable Data File Compression. This is a member of the `MethodFileInfo` object.

**Type**
Read/Write Boolean

**Example**
This example sets the enable data file compression option:

```plaintext
MethodFileInfo.MethodPropEnableDataFileCompress = True
```

**MethodPropertiesDescription Property**
Sets or returns the method properties description. This is a member of the `MethodFileInfo` object.

**Type**
Read/Write String

**Example**
This example sets the method properties description:
MethodFileInfo.MethodPropertiesDescription = "Method Property Description"

**MethodPropNumRepsInRollingAvg Property**

Sets or returns the method properties Number of Replicates in a Rolling Average option. This is a member of the MethodFileInfo object.

**Type**
Read/Write Long (4 byte integer value)

**Example**
This example sets the method properties number of replicates in a rolling average:

```
MethodFileInfo.MethodPropNumRepsInRollingAvg = 3
```

**MethodPropResponseFactorDef Property**

Sets or returns the method properties Response Factor Definition option. This is a member of the MethodFileInfo object.

The response factor definition is one of the following constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Area / Amount.</td>
</tr>
<tr>
<td>1</td>
<td>Amount / Area.</td>
</tr>
</tbody>
</table>

**Type**
Read/Write Long (4 byte integer value)

**Example**
This example sets the method properties response factor definition to be Area / Amount:

```
MethodFileInfo.MethodPropNumRepsInRollingAvg = 0
```

**Name Property**

Returns the name of this object. This is a member of the MethodFileInfo object.

**Type**
Read Only String

**Example**
This example displays a message box with the name of this object to the user:

```
strMsg = "The name of the MethodFileInfo object is " & _
MethodFileInfo.Name
MsgBox (strMsg)
```
**RefreshMethodCustomParams Method**

Refreshes the MethodCustomParams object, which reloads the MethodCustomParams list. This is a member of the MethodFileInfo object.

**Returns**
Void

**Syntax**
RefreshMethodCustomParams()

**Example**
This example displays the most recent list of MethodCustomParams:

```vba
MethodFileInfo.RefreshMethodCustomParams
For Each MethodCustomParam In MethodFileInfo.MethodCustomParams
    MsgBox MethodCustomParam.ParameterName
Next
```

**RemoveAllPDA_MultiChroms Method**

Removes all multi-chromatograms from the PDA multi-chromatogram table. This is a member of the MethodFileInfo object.

**Returns**
Boolean

**Syntax**
RemoveAllPDA_MultiChroms()

**Example**
This example removes all multi-chromatograms from the table:

```vba
Dim bRemoveAllMultiChromsOK As Boolean
bRemoveAllMultiChromsOK = MethodFileInfo.RemoveAllPDA_MultiChroms
```

**RemovePDA_MultiChrom Method**

Removes the specified multi-chromatogram from the PDA multi-chromatogram table. This is a member of the MethodFileInfo object.

**Returns**
Boolean

**Syntax**
RemovePDA_MultiChrom(RowNumber)

  **RowNumber** Required Long (4 byte integer value). This is the multi-chromatogram row number in the multi-chromatogram table. Row numbers start at 1.
Example
This example removes the second multi-chromatogram from the table:

```vbnet
Dim bRemoveMultiChromOK As Boolean
bRemoveMultiChromOK = _
MethodFileInfo.RemovePDA_MultiChrom(2)
```

**SetAfterAnalysisData Method**
Sends all the After Analysis values to the open method. This is a member of the MethodFileInfo object.

**Returns**
Void

**Syntax**
SetAfterAnalysisData()

**Example**
This example sends all the After Analysis properties to the open method:

```
MethodFileInfo.SetAfterAnalysisData
```

**Remarks**
The following properties will be sent to the method using this call:

- AfterAnalysisAdditionalParams
- AfterAnalysisFileName
- UseAfterAnalysisProgram

Any properties not set through automation will retain their current value.

**SetAfterExportData Method**
Sends all the After Export values to the open method. This is a member of the MethodFileInfo object.

**Returns**
Void

**Syntax**
SetAfterExportData()

**Example**
This example sends all the After Export properties to the open method:

```
MethodFileInfo.SetAfterExportData
```

**Remarks**
The following properties will be sent to the method using this call:
AfterExportAdditionalParams
AfterExportFileName
UseAfterExportProgram

Any properties not set through automation will retain their current value.

**SetAuditTrailMode Method**

Sets the audit trail mode for the current method file. This is a member of the MethodFileInfo object.

**Returns**
Boolean

**Syntax**
SetAuditTrailMode (IMode)

**Example**
This example sets the audit trail mode for no prompting:

```
Dim bSuccess as Boolean
bSuccess = MethodFileInfo.SetAuditTrailMode(2)
```

**SetBaselineData Method**

Sends all the Baseline values to the open method. This is a member of the MethodFileInfo object.

**Returns**
Void

**Syntax**
SetBaselineData()

**Example**
This example sends all the Baseline properties to the open method:

```
MethodFileInfo.SetBaselineData
```

**Remarks**
The following properties will be sent to the method using this call:

BaselineFileName
UseBaselineFile

Any properties not set through automation will retain their current value.
**SetBeforeAnalysisData Method**

Sends all the Before Analysis values to the open method. This is a member of the `MethodInfo` object.

**Returns**

Void

**Syntax**

`SetBeforeAnalysisData()`

**Example**

This example sends all the Before Analysis properties to the open method:

```csharp
MethodInfo.SetBeforeAnalysisData
```

**Remarks**

The following properties will be sent to the method using this call:

- BeforeAnalysisAdditionalParams
- BeforeAnalysisFileName
- UseBeforeAnalysisProgram

Any properties **not** set through automation will retain their current value.

**SetBeforeRunData Method**

Sends all the Before Run values to the open method. This is a member of the `MethodInfo` object.

**Returns**

Void

**Syntax**

`SetBeforeRunData()`

**Example**

This example sends all the Before Run properties to the open method:

```csharp
MethodInfo.SetBeforeRunData
```

**Remarks**

The following properties will be sent to the method using this call:

- BeforeRunAdditionalParams
- BeforeRunFileName
- UseBeforeRunProgram

Any properties **not** set through automation will retain their current value.
**SetChannelIDsToSubtract Method**

Sets the Channel IDs to subtract from a baseline program. This is a member of the MethodFileInfo object.

**Returns**

Void

**Syntax**

SetChannelIDsToSubtract vChannelIDValues

  vChannelIDValues Variant. This is a variant that will require an array of String values. Each value in this array represents a channel ID to be subtracted. Any channel IDs not included in this list will be unchecked in the list of channels to be subtracted on the instrument. To uncheck all channel IDs set the variant to Empty.

**Example**

This example sets the list of channel IDs to subtract from a baseline program:

```vbscript
Dim vChannelIds As Variant
MethodFileInfo.SetChannelIDsToSubtract vChannelIds
```

**SetMethodItemValue Method**

Sets instrument specific method item values in the current open method. After setting these values the method needs to be saved to retain the values. Use this method to set any Initial values for a program table even if they appear to be part of the table. This is a member of the MethodFileInfo object.

**Returns**

Long (4 byte integer value)

  0 – Success: Value Set Successfully.
  101 – Failed: Item not supported by device.
  102 – Failed: Value out of range

**Syntax**

SetMethodItemValue(ItemID, ItemValue)

  ItemID Required Long (4 byte integer value). The ID of the method item to set. For a list of ItemIDs, refer to Method Item IDs.
  ItemValue Required Double (8 byte floating point value). The value to set the item.

**Example**

This example sets the Run Time to 12 seconds:

```vbscript
Dim lRet as Long
lRet = MethodFileInfo.SetMethodItemValue(1000, 12.0)
```
**SetMethodOptionsData Method**

Sends all the Method Options values to the open method. This is a member of the `MethodFileInfo` object.

**Returns**
Void

**Syntax**
`SetMethodOptionsData()`

**Example**
This example sends all the Method Options properties to the open method:

```
MethodFileInfo.SetMethodOptionsData
```

**Remarks**
The following properties will be sent to the method using this call:

- `MethodPropAnalysisPeriod`
- `MethodPropAnalyzeAfterAcq`
- `MethodPropAnalyzeDuringAcq`
- `MethodPropAutoAvgReplicates`
- `MethodPropEnableDataFileCompress`
- `MethodPropNumRepsInRollingAvg`
- `MethodPropResponseFactorDef`

Any properties not set through automation will retain their current value.

**SetMethodProgramValue Method**

Sets instrument specific method time program values in the current open method. After setting these values, save the method to retain the values. Use this method to set any program steps that are not Initial initial values. Use the `SetMethodItemValue` method to set Initial values even if they appear to be part of the program table. This is a member of the `MethodFileInfo` object.

**Returns**
Long (4 byte integer value)

- 0 – Success: Value Set Successfully.
- 101 – Failed: Item not supported by device.
- 102 – Failed: Value out of range
- 104 – Failed: Program time out of range
- 105 – Failed: Max program steps exceeded.

**Syntax**
`SetMethodProgramValue (ItemID, Time, ItemValue)`
**ItemID**
Required Long (4 byte integer value). The ID of the method item to set.
For a list of ItemIDs, refer to Method Item IDs.
Run Time, ItemID 1000, is not supported by this method.

**Time**
Required Long (4 byte integer value). The program step time.

**ItemValue**
Required Double (8 byte floating point value). The value to set the item.

**Example**
This example sets % Solvent C to 54.0 at 60 seconds:

```vba
Dim lRet as Long
lRet = MethodFileInfo.SetMethodProgramValue(1012, 60, 54.0)
```

---

**SetPDA_BOOLData Method**

Sets boolean type PDA values. This is a member of the MethodFileInfo object.

**Returns**
Boolean

**Syntax**
SetPDA_BOOLData (ItemID, ItemValue)

**ItemID**
Required Long (4 byte integer value). The ID of the PDA item to set.

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>104121</td>
<td>Spectrum Background Correction Flag</td>
</tr>
<tr>
<td>104122</td>
<td>Spectrum Interpolate Spectra Flag</td>
</tr>
<tr>
<td>104202</td>
<td>Spectrum Similarity Calculation Type Flag</td>
</tr>
<tr>
<td>104203</td>
<td>Spectrum Up Slope Calculation Type Flag</td>
</tr>
<tr>
<td>104204</td>
<td>Spectrum Down Slope Calculation Type Flag</td>
</tr>
<tr>
<td>104205</td>
<td>Spectrum Lambda Max Calculation Type Flag</td>
</tr>
</tbody>
</table>

**ItemValue**
Required Boolean. The value of the item.

**Example**
This example sets Spectrum Background Correction Flag, a PDA boolean type value, to True:

```vba
Dim bSetPDABoolValueOK As Boolean
bSetPDABoolValueOK = _
MethodFileInfo.SetPDA_BOOLData(104121, True)
```

---

**SetPDA_INT32Data Method**

Sets long type PDA values. This is a member of the MethodFileInfo object.

**Returns**
Boolean
Syntax
SetPDA_INT32Data (ItemID, ItemValue)

**ItemID**
Required Long (4 byte integer value). The ID of the PDA item to get.

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>104100</td>
<td>Ratio Bandwidth 1</td>
</tr>
<tr>
<td>104101</td>
<td>Ratio Bandwidth 2</td>
</tr>
<tr>
<td>104102</td>
<td>Ratio Wavelength 1</td>
</tr>
<tr>
<td>104103</td>
<td>Ratio Wavelength 2</td>
</tr>
<tr>
<td>104104</td>
<td>Ratio Threshold</td>
</tr>
<tr>
<td>104120</td>
<td>Spectrum Filter Type</td>
</tr>
<tr>
<td>104125</td>
<td>Spectrum Peak Type</td>
</tr>
<tr>
<td>104126</td>
<td>Spectrum Average N Spectra</td>
</tr>
<tr>
<td>104127</td>
<td>Spectrum Lambda Max From</td>
</tr>
<tr>
<td>104128</td>
<td>Spectrum Lambda Max To</td>
</tr>
</tbody>
</table>

**ItemValue**
Required Long (4 byte integer value). The value of the item.
For the Spectrum Filter Type, the value set needs to be one of the following:

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200000</td>
<td>None</td>
</tr>
<tr>
<td>200001</td>
<td>Smooth</td>
</tr>
<tr>
<td>200002</td>
<td>1st Derivative</td>
</tr>
<tr>
<td>200003</td>
<td>2nd Derivative</td>
</tr>
</tbody>
</table>

For the Spectrum Peak Type, the value set needs to be one of the following:

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>104140</td>
<td>Apex</td>
</tr>
<tr>
<td>104141</td>
<td>Average Up, Apex Down</td>
</tr>
<tr>
<td>104142</td>
<td>Average Every N Spectra</td>
</tr>
</tbody>
</table>

**Example**
This example sets Spectrum Peak Type, a PDA long type value, to Average Up, Apex Down:

```vbnet
Dim bSetPDALongValueOK As Boolean
bSetPDALongValueOK = _
    MethodFileInfo.SetPDA_INT32Data(104125, 104141)
```

**UseAfterAnalysisProgram Property**
Sets or returns the After Analysis flag. This is a member of the `MethodFileInfo` object.

**Type**
Read/Write Boolean

**Example**
This example sets the use After Analysis program flag:
MethodFileInfo.UseAfterAnalysisProgram = True

**UseAfterExportProgram Property**
Sets or returns the After Export flag. This is a member of the `MethodFileInfo` object.

**Type**
Read/Write Boolean

**Example**
This example sets the use After Export program flag:

```
MethodFileInfo.UseAfterExportProgram = True
```

**UseBaselineFile Property**
Sets or returns the Baseline File flag. This is a member of the `MethodFileInfo` object.

**Type**
Read/Write Boolean

**Example**
This example sets the use baseline file flag:

```
MethodFileInfo.UseBaselineFile = True
```

**UseBeforeAnalysisProgram Property**
Sets or returns the Before Analysis flag. This is a member of the `MethodFileInfo` object.

**Type**
Read/Write Boolean

**Example**
This example sets the use Before Analysis program flag:

```
MethodFileInfo.UseBeforeAnalysisProgram = True
```

**UseBeforeRunProgram Property**
Sets or returns the Before Run flag. This is a member of the `MethodFileInfo` object.

**Type**
Read/Write Boolean

**Example**
This example sets the use Before Run program flag:

```
MethodFileInfo.UseBeforeRunProgram = True
```
**MethodCustomParams Object**

This object contains a collection of all the [MethodCustomParam](#) objects that are currently available. All defined method custom parameters will be available in the collection. The [MethodFileInfo](#) object returns a pointer to this object. Applications can enumerate through the collection of method custom parameters by using For … Next statements or by using For … Each statements. For an overview of all objects, see the [Object Hierarchy](#).

**Example**

This example displays the name of each method custom parameter in the collection using For … Each statements:

```vba
For Each MethodCustomParam In MethodCustomParams
    MsgBox ("Method Custom Parameter Name: " & _
            MethodCustomParam.ParameterName)
Next
```

This example displays the name of each method custom parameter in the collection using For … Next statements:

```vba
nCount = MethodCustomParams.Count
For nIndex = 0 To nCount - 1
    MsgBox ("Method Custom Parameter Name: " & _
            MethodCustomParams(nIndex).ParameterName)
Next nIndex
```

**Count Property**

Returns the number of method custom parameters available. This is a member of the [MethodCustomParams](#) object.

**Type**

Read Only Long (4 byte integer value)

**Example**

This example displays the name of each method custom parameter to the user:

```vba
nCount = MethodCustomParams.Count
For nIndex = 0 To nCount - 1
    MsgBox ("Method Custom Parameter Name: " & _
            MethodCustomParams(nIndex).ParameterName)
Next nIndex
```

**Item Property**

Returns the [MethodCustomParam](#) object at the given indexed location. This is a member of the [MethodCustomParams](#) object.
Type
Read Only Object

Syntax
Item(Index)

Index Required Long (4 byte integer value). This is a zero based index that specifies the MethodCustomParam object to be returned.

Example
This example displays the name of each method custom parameter to the user:

```
nCount = MethodCustomParams.Count
For nIndex = 0 To nCount - 1
    MsgBox ("Method Custom Parameter Name: " & MethodCustomParams(nIndex).ParameterName)
Next nIndex
```

Remarks
Item is the default member of this collection, so the following two lines of code are equivalent:

```
MethodCustomParams.Item(1)
MethodCustomParams(1)
```

Name Property

Returns the name of this object. This is a member of the MethodCustomParams object.

Type
Read Only String

Example
This example displays a message box with the name of this object to the user:

```
strMsg = "The name of the MethodCustomParams object is " & MethodCustomParams.Name
MsgBox (strMsg)
```
**MethodCustomParam Object**

This object represents a method custom parameter. The `MethodCustomParams` collection object contains all of the method custom parameter objects that are currently available. For an overview of all objects, see the Object Hierarchy.

**AdditionalParams Property**

Sets or returns the additional parameters for the method custom parameter. This is a member of the `MethodCustomParam` object.

**Type**

Read/Write String

**Example**

This example displays a message box with the additional parameters for the method custom parameter:

```vba
strMsg = "The additional parameters of the current " & "method custom parameter are " & _
MethodCustomParam.AdditionalParams
MsgBox (strMsg)
```

**InUse Property**

Sets or returns whether the method custom parameter is used or not. This is a member of the `MethodCustomParam` object.

**Type**

Read/Write Boolean

**Example**

This example displays a message box with the use status for the method custom parameter:

```vba
strMsg = "The use status of the current " & "method custom parameter is " & _
MethodCustomParam.InUse
MsgBox (strMsg)
```

**ParameterID Property**

Returns the method custom parameter ID. This is a member of the `MethodCustomParam` object.
Type
Read Only Long

Example
This example displays the ID of each method custom parameter to the user:

For Each MethodCustomParam In MethodCustomParams
    MsgBox ("Method Custom Parameter ID: " & MethodCustomParam.ParameterID)
Next

ParameterName Property

Returns the method custom parameter name. This is a member of the MethodCustomParam object.

Type
Read Only String

NOTE: This has become a Read Only String because changing a custom parameter name causes a new custom parameter to be created. Use the AddNewCustomParam method of the Instrument interface to create a new custom parameter. In other words, a custom parameter name cannot be changed once it is created. You must delete a custom parameter with an unwanted name and create a new custom parameter with the correct name.

Example
This example displays the name of each method custom parameter to the user:

For Each MethodCustomParam In MethodCustomParams
    MsgBox ("Method Custom Parameter Name: " & MethodCustomParam.ParameterName)
Next

ParamType Property

Sets or returns the method custom parameter type. This is a member of the MethodCustomParam object.

The method custom parameter type is one of the following constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Per-peak</td>
</tr>
<tr>
<td>2</td>
<td>System-wide</td>
</tr>
</tbody>
</table>

Type
Read/Write Long (4 byte integer value)

Example
This example sets the method custom parameter type to be per peak:
**ReturnType Property**

Sets or returns the method custom parameter return type. This is a member of the `MethodCustomParam` object.

The method custom parameter return type is one of the following constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Numeric</td>
</tr>
<tr>
<td>2</td>
<td>String</td>
</tr>
</tbody>
</table>

**Type**

Read/Write Long (4 byte integer value)

**Example**

This example sets the method custom parameter return type to be a string:

```csharp
MethodCustomParam.ReturnType = 2
```

**SetMethodCustomParamData Method**

Sends all the Method Custom Param values to the open method. This is a member of the `MethodCustomParam` object.

**Returns**

Void

**Syntax**

`SetMethodCustomParamData()`

**Example**

This example sends all the Method Custom Param properties to the open method:

```csharp
MethodInfo.SetMethodCustomParamData
```

**Remarks**

The following properties will be sent to the method using this call:

- `AdditionalParams`
- `InUse`
- `ParameterName`
- `ParamType`
ReturnType
SourceFile

Any properties not set through automation will retain their current value.

**SourceFile Property**

Sets or returns the source file name for the method custom parameter. This is a member of the `MethodCustomParam` object.

**Type**
Read/Write String

**Example**
This example displays a message box with the source file name for the method custom parameter:

```vbscript
strMsg = "The source file of the current " & "method custom parameter is " & MethodCustomParam.SourceFile
MsgBox (strMsg)
```
MethodChannels Object

This object contains a collection of all the MethodChannel objects that are currently available. All defined method channels will be available in the collection. The Instrument object returns a pointer to this object. Applications can enumerate through the collection of method channels by using For ... Next statements or by using For ... Each statements. For an overview of all objects, see the Object Hierarchy.

Example
This example displays the column description of each method channel in the collection using For ... Each statements:

    For Each MethodChannel In MethodChannels
        MsgBox ("Method channel column description is: " & _
            MethodChannel.ColumnDescription)
    Next

This example displays the column description of each method channel in the collection using For ... Next statements:

    nCount = MethodChannels.Count
    For nIndex = 0 To nCount - 1
        MsgBox ("Method channel column description is: " & _
            MethodChannels(nIndex).ColumnDescription)
    Next nIndex

Count Property

Returns the number of method channels available. This is a member of the MethodChannels object.

Type
Read Only Long (4 byte integer value)

Example
This example displays the column description of each method channel to the user:

    nCount = MethodChannels.Count
    For nIndex = 0 To nCount - 1
        MsgBox ("Method channel column description is: " & _
            MethodChannels(nIndex).ColumnDescription)
    Next nIndex
**Item Property**

Returns the `MethodChannel` object at the given indexed location. This is a member of the `MethodChannels` object.

**Type**
Read Only Object

**Syntax**
*Item(Index)*

Index Required Long (4 byte integer value). This is a zero based index that specifies the MethodChannel object to be returned.

**Example**
This example displays the column description of each method channel to the user:

```vbnet
nCount = MethodChannels.Count
For nIndex = 0 To nCount - 1
    MsgBox ("Method channel column description is: " & 
             MethodChannels(nIndex).ColumnDescription)
Next nIndex
```

**Remarks**
Item is the default member of this collection, so the following two lines of code are equivalent:

```vbnet
MethodChannels.Item(1)
MethodChannels(1)
```

**Name Property**

Returns the name of this object. This is a member of the `MethodChannels` object.

**Type**
Read Only String

**Example**
This example displays a message box with the name of this object to the user:

```vbnet
strMsg = "The name of the MethodChannels object is " & _
        MethodChannels.Name
MsgBox (strMsg)
```
**MethodChannel Object**

This object represents a method channel. The MethodChannels collection object contains all of the method channel objects that are currently available. For an overview of all objects, see the Object Hierarchy.

**CalcPerformanceParams Property**

Sets or returns the calculate performance parameters flag for the method channel. This is a member of the MethodChannel object.

**Type**
Read/Write Boolean

**Example**
This example displays a message box with the status of the calculate performance parameters flag for the method channel:

```vba
strMsg = "The calculate performance params flag " _
    & "of the current method channel is set to " & _
    MethodChannel.CalcPerformanceParams
MsgBox (strMsg)
```

**CalcUsingAOH Property**

Sets or returns the calculate using AOH flag for the method channel. This is a member of the MethodChannel object.

**Type**
Read/Write Boolean

**Example**
This example displays a message box with the status of the calculate using AOH flag for the method channel:

```vba
strMsg = "The calculate using AOH flag " _
    & "of the current method channel is set to " & _
    MethodChannel.CalcUsingAOH
MsgBox (strMsg)
```

**CalcUsingDAB_BP_EP_ASTM Property**

Sets or returns the calculate using DAB, BP, EP, or ASTM flag for the method channel. This is a member of the MethodChannel object.
Type
Read/Write Boolean

Example
This example displays a message box with the status of the calculate using DAB, BP, EP, or ASTM flag for the method channel:

```vba
strMsg = "The calculate using DAB, BP, EP, or ASTM flag " & "of the current method channel is set to " & _
MethodChannel.CalcUsingDAB_BP_EP_ASTM
MsgBox (strMsg)
```

**CalcUsingEMG Property**

Sets or returns the calculate using EMG flag for the method channel. This is a member of the `MethodChannel` object.

Type
Read/Write Boolean

Example
This example displays a message box with the status of the calculate using EMG flag for the method channel:

```vba
strMsg = "The calculate using EMG flag " & "of the current method channel is set to " & _
MethodChannel.CalcUsingEMG
MsgBox (strMsg)
```

**CalcUsingJP Property**

Sets or returns the calculate using JP flag for the method channel. This is a member of the `MethodChannel` object.

Type
Read/Write Boolean

Example
This example displays a message box with the status of the calculate using JP flag for the method channel:

```vba
strMsg = "The calculate using JP flag " & "of the current method channel is set to " & _
MethodChannel.CalcUsingJP
MsgBox (strMsg)
```

**CalcUsingUSP Property**

Sets or returns the calculate using USP flag for the method channel. This is a member of the `MethodChannel` object.
**Type**
Read/Write Boolean

**Example**
This example displays a message box with the status of the calculate using USP flag for the method channel:

```vba
strMsg = "The calculate using USP flag " & "of the current method channel is set to " & MethodChannel.CalcUsingUSP
MsgBox (strMsg)
```

**Channel_ID Property**
Returns the channel ID of this method channel. This is a member of the `MethodChannel` object.

**Type**
Read Only Long (4 byte integer value)

**Example**
This example displays a message box with the channel ID of the method channel:

```vba
strMsg = "The channel ID for the current " & "method channel is " & MethodChannel.Channel_ID
MsgBox (strMsg)
```

**ColumnDescription Property**
Sets or returns the column description for the method channel. This is a member of the `MethodChannel` object.

**Type**
Read/Write String

**Example**
This example displays a message box with the column description for the method channel:

```vba
strMsg = "The column description for the current " & "method channel is " & MethodChannel.ColumnDescription
MsgBox (strMsg)
```

**ColumnInstallationDate Property**
Sets or returns the column installation date for the method channel. This is a member of the `MethodChannel` object.
**Type**  
Read/Write String

**Example**  
This example displays a message box with the column installation date for the method channel:

```vba
strMsg = "The column installation date for the current " & "method channel is " & MethodChannel.ColumnInstallationDate
MsgBox (strMsg)
```

**ColumnLength Property**  
Sets or returns the column length for the method channel. This is a member of the MethodChannel object.

**Type**  
Read/Write Double (8 byte floating point value)

**Example**  
This example displays a message box with the column length for the method channel:

```vba
strMsg = "The column length for the current " & "method channel is " & MethodChannel.ColumnLength
MsgBox (strMsg)
```

**ColumnSerialNum Property**  
Sets or returns the column serial number for the method channel. This is a member of the MethodChannel object.

**Type**  
Read/Write String

**Example**  
This example displays a message box with the column serial number for the method channel:

```vba
strMsg = "The column serial number for the current " & "method channel is " & MethodChannel.ColumnSerialNum
MsgBox (strMsg)
```

**ParticleDiameter Property**  
Sets or returns the particle diameter for the method channel. This is a member of the MethodChannel object.

**Type**
Read/Write Double (8 byte floating point value)

Example
This example displays a message box with the particle diameter for the method channel:

```vba
strMsg = "The particle diameter for the current " & 
MethodChannel.ParticleDiameter & 
" method channel is " & 
MsgBox (strMsg)
```

**SetMethodPerformanceParamsData Method**

Sends all the Method Performance Params values to the open method. This is a member of the `MethodChannel` object.

Returns
Void

Syntax
SetMethodPerformanceParamsData()

Example
This example sends all the Method Custom Param properties to the open method:

```vba
MethodFileInfo.SetMethodPerformanceParamsData
```

Remarks
The following properties will be sent to the method using this call:

- CalcPerformanceParams
- CalcUsingAOH
- CalcUsingDAB_BP_EP_ASTM
- CalcUsingEMG
- CalcUsingJP
- CalcUsingUSP
- ColumnDescription
- ColumnInstallationDate
- ColumnLength
- ColumnSerialNum
- ParticleDiameter
- UnretainedPeakTime

Any properties not set through automation will retain their current value.

**UnretainedPeakTime Property**

Sets or returns the unretained peak time for the method channel. This is a member of the `MethodChannel` object.
Type
Read/Write Double (8 byte floating point value)

Example
This example displays a message box with the unretained peak time for the method channel:

```vbnet
strMsg = "The unretained peak time for the current " & MethodChannel.UnretainedPeakTime
MsgBox (strMsg)
```
**MethodTraces Object**

This object contains a collection of all the **MethodTrace** objects that are currently available. All defined method traces will be available in the collection. The **Instrument** object returns a pointer to this object. Applications can enumerate through the collection of method traces by using For … Next statements or by using For … Each statements. For an overview of all objects, see the Object Hierarchy.

**Example**

This example displays the name of each method trace in the collection using For … Each statements:

```vba
For Each MethodTrace In MethodTraces
    MsgBox ("Method Trace Name: " & _
             MethodTrace.TraceName)
Next
```

This example displays the name of each method trace in the collection using For … Next statements:

```vba
nCount = MethodTraces.Count
For nIndex = 0 To nCount - 1
    MsgBox ("Method Trace Name: " & _
             MethodTraces(nIndex).TraceName)
Next nIndex
```

**Count Property**

Returns the number of method traces available. This is a member of the **MethodTraces** object.

**Type**

Read Only Long (4 byte integer value)

**Example**

This example displays the name of each method trace to the user:

```vba
nCount = MethodTraces.Count
For nIndex = 0 To nCount - 1
    MsgBox ("Method Trace Name: " & _
             MethodTraces(nIndex).TraceName)
Next nIndex
```

**Item Property**

Returns the **MethodTrace** object at the given indexed location. This is a member of the **MethodTraces** object.

**Type**

Read Only Object
Syntax
Item(Index)

Index Required Long (4 byte integer value). This is a zero based index that specifies the MethodTrace object to be returned.

Example
This example displays the name of each method trace to the user:

nCount = MethodTraces.Count
For nIndex = 0 To nCount - 1
    MsgBox ("Method Trace Name: " & _
        MethodTraces(nIndex).TraceName)
Next nIndex

Remarks
Item is the default member of this collection, so the following two lines of code are equivalent:

    MethodTraces.Item(1)
    MethodTraces(1)

Name Property
Returns the name of this object. This is a member of the MethodTraces object.

Type
Read Only String

Example
This example displays a message box with the name of this object to the user:

    strMsg = "The name of the MethodTraces object is " & _
        MethodTraces.Name
    MsgBox (strMsg)
MethodTrace Object

This object represents a method trace. The MethodTraces collection object contains all of the method trace objects that are currently available. For an overview of all objects, see the Object Hierarchy.

AddNewNamedPeak Method

Adds a new named peak to the named peak list. This is a member of the MethodTrace object.

Returns
Boolean

Syntax
AddNewCustomParam

Example
This example adds a new named peak:

    Dim bCreatedNamedPeakOK As Boolean
    bCreatedNamedPeakOK = MethodTrace.AddNewNamedPeak

NOTE:
Use the NamedPeak interface to set the new named peak's properties. Then call the SetNamedPeakData method.

Channel_ID Property

Returns the channel ID associated with this method trace. This is a member of the MethodTrace object.

Type
Read Only Long (4 byte integer value)

Example
This example displays a message box with the channel ID associated with the method trace:

    strMsg = "The channel ID for the current " & "method trace is " & MethodTrace.Channel_ID
    MsgBox (strMsg)

DeleteNamedPeak Method

Deletes a named peak from the named peak list. This is a member of the MethodTrace object.
Returns
Boolean

Syntax
DeleteCustomParam(PeakID)

PeakID    Required Long (4 byte integer value). This is the ID of the named peak to be deleted.

Example
This example deletes a named peak:

    Dim bDeletedNamedPeakOK As Boolean
    bDeletedNamedPeakOK = MethodTrace.DeleteNamedPeak(3)

ExportPath Property
Sets or returns the export path for the method trace. This is a member of the MethodTrace object.

Type
Read/Write String

Example
This example displays a message box with the export path for the method trace:

    strMsg = "The export path for the current " 
             & "method trace is " & MethodTrace.ExportPath
    MsgBox (strMsg)

IntegrationEvents Property
Returns the IntegrationEvents object. This is a member of the MethodTrace object.

Type
Read Only Object

Example
This example displays the event type of each integration event to the user:

    For Each IntegrationEvent In MethodTrace.IntegrationEvents
        MsgBox ("Event Type: " & IntegrationEvent.EventType)
    Next

NamedPeaks Property
Returns the NamedPeaks object. This is a member of the MethodTrace object.

Type
Read Only Object

Example
This example displays the name of each named peak to the user:

For Each NamedPeak In MethodTrace.NamedPeaks
    MsgBox ("Peak Name: " & NamedPeak.PeakName)
Next

RefreshIntegrationEvents Method
Refreshes the IntegrationEvents object, which reloads the IntegrationEvents list. This is a member of the MethodTrace object.

Returns
Void

Syntax
RefreshIntegrationEvents()

Example
This example displays the most recent list of IntegrationEvents:

    MethodTrace.RefreshIntegrationEvents
    For Each IntegrationEvent In MethodTrace.IntegrationEvents
        MsgBox IntegrationEvent.EventType
    Next

RefreshNamedPeaks Method
Refreshes the NamedPeaks object, which reloads the NamedPeaks list. This is a member of the MethodTrace object.

Returns
Void

Syntax
RefreshNamedPeaks()

Example
This example displays the most recent list of NamedPeaks:

    MethodTrace.RefreshNamedPeaks
    For Each NamedPeak In MethodTrace.NamedPeaks
        MsgBox NamedPeak.PeakName
    Next
**TraceID Property**

As of Elite 3.2, TraceID is no longer supported. If used, either -1 or an OLE exception, AFX_IDP_GET_NOT_SUPPORTED (0xF18C), will be returned. It has been replaced by the TraceIDEx property.

Returns the trace ID of this method trace. This is a member of the MethodTrace object. This function no longer works, starting with EZChrom Elite 3.2.

**Type**
Read Only Long (4 byte integer value)

**Example**
This example displays a message box with the trace ID of the method trace:

```vba
strMsg = "The trace ID for the current " _ & "method trace is " & MethodTrace.TraceID
MsgBox (strMsg)
```

**TraceIDEx Property**

Returns the trace ID of this method trace. This is a member of the MethodTrace object. For older code this needs to replace TraceID calls.

**Type**
Read Only string

**Example**
This example displays a message box with the trace ID of the method trace:

```vba
Dim strTraceID as String
Dim strMsg as String

strTraceID = MethodTrace.TraceIDEx
strMsg = "The trace ID for the current " _ & "method trace is " & strTraceID
MsgBox strMsg
```

**TraceName Property**

Returns the name of this method trace. This is a member of the MethodTrace object.

**Type**
Read Only String

**Example**
This example displays a message box with the name of the method trace:
strMsg = "The name of the current " & "method trace is " & MethodTrace.TraceName
MsgBox (strMsg)
**IntegrationEvents Object**

This object contains a collection of all the IntegrationEvent objects that are currently available. All defined integration events will be available in the collection. The MethodTrace object returns a pointer to this object. Applications can enumerate through the collection of integration events by using For … Next statements or by using For … Each statements. For an overview of all objects, see the Object Hierarchy.

**Example**
This example displays the event type of each integration event in the collection using For … Each statements:

```vba
For Each IntegrationEvent In IntegrationEvents
    MsgBox ("Event Type: " & IntegrationEvent.EventType)
Next
```

This example displays the event type of each integration event in the collection using For … Next statements:

```vba
nCount = IntegrationEvents.Count
For nIndex = 0 To nCount - 1
    MsgBox ("Event Type: " & IntegrationEvents(nIndex).EventType)
Next nIndex
```

**Count Property**
Returns the number of integration events available. This is a member of the IntegrationEvents object.

**Type**
Read Only Long (4 byte integer value)

**Example**
This example displays the event type of each integration event to the user:

```vba
nCount = IntegrationEvents.Count
For nIndex = 0 To nCount - 1
    MsgBox ("Event Type: " & IntegrationEvents(nIndex).EventType)
Next nIndex
```

**Item Property**
Returns the IntegrationEvent object at the given indexed location. This is a member of the IntegrationEvents object.
Type
Read Only Object

Syntax
Item(Index)

   Index       Required Long (4 byte integer value). This is a zero based index that specifies the IntegrationEvent object to be returned.

Example
This example displays the event type of each integration event to the user:

   nCount = IntegrationEvents.Count
   For nIndex = 0 To nCount - 1
      MsgBox ("Event Type: " & IntegrationEvents(nIndex).EventType)
   Next nIndex

Remarks
Item is the default member of this collection, so the following two lines of code are equivalent:

   IntegrationEvents.Item(1)
   IntegrationEvents(1)

Name Property

Returns the name of this object. This is a member of the IntegrationEvents object.

Type
Read Only String

Example
This example displays a message box with the name of this object to the user:

   strMsg = "The name of the IntegrationEvents object is " & IntegrationEvents.Name
   MsgBox (strMsg)
IntegrationEvent Object

This object represents an integration event. The `IntegrationEvents` collection object contains all of the integration event objects that are currently available. For an overview of all objects, see the Object Hierarchy.

NOTE: Currently integration event data may only be read from the instrument. The functionality to set data through automation will be added in the future.

EventType Property

Returns the event type of this integration event. This is a member of the `IntegrationEvent` object.

The event type is one of the following constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Unknown</td>
</tr>
<tr>
<td>0</td>
<td>Integration Off</td>
</tr>
<tr>
<td>1</td>
<td>Threshold</td>
</tr>
<tr>
<td>2</td>
<td>Shoulder Sensitivity</td>
</tr>
<tr>
<td>3</td>
<td>Width</td>
</tr>
<tr>
<td>4</td>
<td>Valley to Valley</td>
</tr>
<tr>
<td>5</td>
<td>Horizontal Baseline</td>
</tr>
<tr>
<td>6</td>
<td>Tangent Skim</td>
</tr>
<tr>
<td>7</td>
<td>Minimum Area</td>
</tr>
<tr>
<td>8</td>
<td>Negative Peak</td>
</tr>
<tr>
<td>9</td>
<td>Manual Baseline</td>
</tr>
<tr>
<td>10</td>
<td>Manual Peak</td>
</tr>
<tr>
<td>11</td>
<td>Front Tangent Skim</td>
</tr>
<tr>
<td>12</td>
<td>Split Peak</td>
</tr>
<tr>
<td>13</td>
<td>Backward Horizontal Baseline</td>
</tr>
<tr>
<td>14</td>
<td>Disable End Peak Detection</td>
</tr>
<tr>
<td>15</td>
<td>Force Peak Start</td>
</tr>
<tr>
<td>16</td>
<td>Force Peak Stop</td>
</tr>
<tr>
<td>17</td>
<td>Move BL Baseline Start</td>
</tr>
<tr>
<td>18</td>
<td>Move BL Baseline Stop</td>
</tr>
<tr>
<td>19</td>
<td>Re-assign Peak</td>
</tr>
<tr>
<td>20</td>
<td>Reset Baseline</td>
</tr>
<tr>
<td>21</td>
<td>Reset Baseline at Valley</td>
</tr>
<tr>
<td>22</td>
<td>Lowest Point Horizontal Baseline</td>
</tr>
</tbody>
</table>

Type

Read Only Long (4 byte integer value)

Example

This example gets the event type:

```vba
strMsg = "The event type of the " & " current integration event is " & _
        IntegrationEvent.EventType
MsgBox (strMsg)
```
**EventUsed Property**

Returns the event used flag. This is a member of the `IntegrationEvent` object.

**Type**
Read Only Boolean

**Example**
This example displays a message box with the status of the event used flag for the integration event:

```vba
strMsg = "The event used flag of the " & " current integration event is set to " & _
IntegrationEvent.EventUsed
MsgBox (strMsg)
```

**StartTime Property**

Returns the start time for the integration event. This is a member of the `IntegrationEvent` object.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the start time for the integration event:

```vba
strMsg = "The start time for the current " & _
"integration event is " & _ IntegrationEvent.StartTime
MsgBox (strMsg)
```

**StopTime Property**

Returns the stop time for the integration event. This is a member of the `IntegrationEvent` object.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the stop time for the integration event:

```vba
strMsg = "The stop time for the current " & _
"integration event is " & _ IntegrationEvent.StopTime
MsgBox (strMsg)
```
**Value Property**

Returns the value for the integration event. This is a member of the `IntegrationEvent` object.

**Type**
Read Only Double (8 byte floating point value)

**Example**
This example displays a message box with the value for the integration event:

```vbnet
strMsg = "The value for the current " & "integration event is " & _ IntegrationEvent.Value
MsgBox (strMsg)
```
NamedPeaks Object

This object contains a collection of all the NamedPeak objects that are currently available. All defined named peaks will be available in the collection. The MethodTrace object returns a pointer to this object. Applications can enumerate through the collection of named peaks by using For … Next statements or by using For … Each statements. For an overview of all objects, see the Object Hierarchy.

Example
This example displays the name of each named peak in the collection using For … Each statements:

```vba
For Each NamedPeak In NamedPeaks
    MsgBox "Peak Name: " & NamedPeak.PeakName
Next
```

This example displays the name of each named peak in the collection using For … Next statements:

```vba
nCount = NamedPeaks.Count
For nIndex = 0 To nCount - 1
    MsgBox "Peak Name: " & 
        NamedPeaks(nIndex).PeakName
Next nIndex
```

Count Property

Returns the number of integration events available. This is a member of the NamedPeaks object.

**Type**
Read Only Long (4 byte integer value)

**Example**
This example displays the name of each named peak to the user:

```vba
nCount = NamedPeaks.Count
For nIndex = 0 To nCount - 1
    MsgBox "Peak Name: " & 
        NamedPeaks(nIndex).PeakName
Next nIndex
```

Item Property

Returns the NamedPeak object at the given indexed location. This is a member of the NamedPeaks object.

**Type**
Read Only Object

**Syntax**
**Item(Index)**

**Index** Required Long (4 byte integer value). This is a zero based index that specifies the NamedPeak object to be returned.

**Example**
This example displays the name of the named peaks to the user:

```vba
define nCount = NamedPeaks.Count
For nIndex = 0 To nCount - 1
    MsgBox ("Peak Name: \" & _
        NamedPeaks(nIndex).PeakName)
Next nIndex
```

**Remarks**
Item is the default member of this collection, so the following two lines of code are equivalent:

```vba
NamedPeaks.Item(1)
NamedPeaks(1)
```

**Name Property**
Returns the name of this object. This is a member of the NamedPeaks object.

**Type**
Read Only String

**Example**
This example displays a message box with the name of this object to the user:

```vba
strMsg = "The name of the NamedPeaks object is \" & _
    NamedPeaks.Name
MsgBox (strMsg)
```
NamedPeak Object

This object represents a named peak. The NamedPeaks collection object contains all of the named peak objects that are currently available. For an overview of all objects, see the Object Hierarchy.

CalibFlag Property

Sets or returns the calibration type of this named peak. This is a member of the NamedPeak object.

The calibration type is one of the following constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Replace</td>
</tr>
<tr>
<td>1</td>
<td>Weighted Average</td>
</tr>
</tbody>
</table>

Type
Read/Write Long (4 byte integer value)

Example
This example sets the calibration type to be weighted average:

```
NamedPeak.CalibFlag = 1
```

CalibWeight Property

Sets or returns the calibration weight of this named peak. This is a member of the NamedPeak object.

Type
Read/Write Double (8 byte floating point value)

Example
This example sets the calibration weight:

```
NamedPeak.CalibWeight = 1.5
```

CheckStdConc1 Property

Sets or returns the standard concentration #1 of this named peak. This is a member of the NamedPeak object.

Type
Read/Write Double (8 byte floating point value)

Example
This example sets a standard concentration:

```plaintext
NamedPeak.CheckStdConc1 = 2.7
```

**CheckStdConc2 Property**

Sets or returns the standard concentration #2 of this named peak. This is a member of the `NamedPeak` object.

**Type**
Read/Write Double (8 byte floating point value)

**Example**
This example sets a standard concentration:

```plaintext
NamedPeak.CheckStdConc2 = 3.5
```

**CheckStdConc3 Property**

Sets or returns the standard concentration #3 of this named peak. This is a member of the `NamedPeak` object.

**Type**
Read/Write Double (8 byte floating point value)

**Example**
This example sets a standard concentration:

```plaintext
NamedPeak.CheckStdConc3 = 9.3
```

**CheckStdConc4 Property**

Sets or returns the standard concentration #4 of this named peak. This is a member of the `NamedPeak` object.

**Type**
Read/Write Double (8 byte floating point value)

**Example**
This example sets a standard concentration:

```plaintext
NamedPeak.CheckStdConc4 = 1.2
```
**CheckStdConc5 Property**
Sets or returns the standard concentration #5 of this named peak. This is a member of the `NamedPeak` object.

**Type**
Read/Write Double (8 byte floating point value)

**Example**
This example sets a standard concentration:

```csharp
NamedPeak.CheckStdConc5 = 8.4
```

**CheckStdRD1 Property**
Sets or returns the standard RD #1 of this named peak. This is a member of the `NamedPeak` object.

**Type**
Read/Write Double (8 byte floating point value)

**Example**
This example sets a standard RD:

```csharp
NamedPeak.CheckStdRD1 = 6.7
```

**CheckStdRD2 Property**
Sets or returns the standard RD #2 of this named peak. This is a member of the `NamedPeak` object.

**Type**
Read/Write Double (8 byte floating point value)

**Example**
This example sets a standard RD:

```csharp
NamedPeak.CheckStdRD2 = 8.2
```

**CheckStdRD3 Property**
Sets or returns the standard RD #3 of this named peak. This is a member of the `NamedPeak` object.

**Type**
Read/Write Double (8 byte floating point value)
Example
This example sets a standard RD:

```
NamedPeak.CheckStdRD3 = 3.9
```

**CheckStdRD4 Property**
Sets or returns the standard RD #4 of this named peak. This is a member of the NamedPeak object.

**Type**
Read/Write Double (8 byte floating point value)

**Example**
This example sets a standard RD:

```
NamedPeak.CheckStdRD4 = 5.1
```

**CheckStdRD5 Property**
Sets or returns the standard RD #5 of this named peak. This is a member of the NamedPeak object.

**Type**
Read/Write Double (8 byte floating point value)

**Example**
This example sets a standard RD:

```
NamedPeak.CheckStdRD5 = 1.3
```

**ConcUnits Property**
Sets or returns the concentration units of this named peak. This is a member of the NamedPeak object.

**Type**
Read/Write String

**Example**
This example displays a message box with the concentration units for the named peak:

```
strMsg = "The concentration units for the current " _  
& "named peak are " & NamedPeak.ConcUnits  
MsgBox (strMsg)
```
**DetectMode Property**

Sets or returns the detection mode of this named peak. This is a member of the `NamedPeak` object.

The detection mode is one of the following constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Ret. time</td>
</tr>
<tr>
<td>1</td>
<td>Ret. time with spectral confirm</td>
</tr>
</tbody>
</table>

**Type**

Read/Write Long (4 byte integer value)

**Example**

This example sets the detection mode to be Ret. time:

```vbnet
NamedPeak.DetectMode = 0
```

**DupRDLimit Property**

Sets or returns the duplicate RD limit of this named peak. This is a member of the `NamedPeak` object.

**Type**

Read/Write Double (8 byte floating point value)

**Example**

This example sets the duplicate RD limit:

```vbnet
NamedPeak.DupRDLimit = 2.0
```

**ExpectedRT Property**

Sets or returns the expected RT of this named peak in seconds. This is a member of the `NamedPeak` object.

**Type**

Read/Write Double (8 byte floating point value)

**Example**

This example displays a message box with the expected RT for the named peak:

```vbnet
strMsg = "The expected RT for the current " _
        & "named peak is " & NamedPeak.ExpectedRT
MsgBox (strMsg)
```
**ExpectedRTWindow Property**

Sets or returns the expected RT window of this named peak in seconds. This is a member of the NamedPeak object.

**Type**
Read/Write Double (8 byte floating point value)

**Example**
This example displays a message box with the expected RT window for the named peak:

```vba
strMsg = "The expected RT window for the current " _
        & "named peak is " & NamedPeak.ExpectedRTWindow
MsgBox (strMsg)
```

**FitType Property**

Sets or returns the fit type of this named peak. This is a member of the NamedPeak object.

The fit type is one of the following constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Point-to-Point</td>
</tr>
<tr>
<td>1</td>
<td>Linear</td>
</tr>
<tr>
<td>2</td>
<td>Quadratic</td>
</tr>
<tr>
<td>3</td>
<td>Cubic</td>
</tr>
<tr>
<td>4</td>
<td>Average RF</td>
</tr>
</tbody>
</table>

**Type**
Read/Write Long (4 byte integer value)

**Example**
This example sets the fit type to be cubic:

```vba
NamedPeak.FitType = 3
```

**ForceThruZero Property**

Sets or returns the force thru zero flag of this named peak. This is a member of the NamedPeak object.

**Type**
Read/Write Boolean

**Example**
This example displays a message box with the status of the force thru zero flag for the named peak:

```vba
strMsg = "The force thru zero flag for the current " _
        & "named peak is set to " & _ NamedPeak.ForceThruZero
MsgBox (strMsg)
```
**GetLevelData Method**

Returns the level value for the given level number for the named peak. This is a member of the *NamedPeak* object.

**Returns**

Boolean

NOTE: Return value indicates whether the Level is enabled or not.

FALSE = disabled

TRUE   = enabled

**Syntax**

GetLevelData(LevelNumber, LevelValue)

<table>
<thead>
<tr>
<th>LevelNumber</th>
<th>Long (4 byte integer value). This is the level number that the user wishes to get a value for.</th>
</tr>
</thead>
<tbody>
<tr>
<td>LevelValue</td>
<td>Single (4 byte floating point value). This is the level value that will be returned if the level is enabled.</td>
</tr>
</tbody>
</table>

**Example**

This example requests the level value for level 1 of the named peak:

```vba
Dim sglLevel As Single
Dim bLevelUsed As Boolean

bLevelUsed = NamedPeak.GetLevelData(1, sglLevel)
```

**GetManualRF Method**

Returns the Manual RF value for the named peak. This is a member of the *NamedPeak* object.

**Returns**

Boolean

NOTE: Return value indicates whether Manual RF is enabled or not.

FALSE = disabled

TRUE   = enabled

**Syntax**

GetManualRF(ManualRFValue)

| ManualRFValue     | Double (8 byte floating point value). This is the Manual RF value that will be returned. |

**Example**

This example requests the Manual RF value for the named peak:
Dim dManualRF As Double
Dim bManualRFUsed As Boolean


**HighConc Property**
Sets or returns the high concentration value of this named peak. This is a member of the NamedPeak object.

**Type**
Read/Write Double (8 byte floating point value)

**Example**
This example displays a message box with the high concentration value for the named peak:

```
strMsg = "The high concentration value for the current " & 
        "named peak is " & NamedPeak.HighConc
MsgBox (strMsg)
```

**HighSpikeLimit Property**
Sets or returns the high spike limit of this named peak. This is a member of the NamedPeak object.

**Type**
Read/Write Double (8 byte floating point value)

**Example**
This example displays a message box with the high spike limit for the named peak:

```
strMsg = "The high spike limit for the current " & 
        "named peak is " & NamedPeak.HighSpikeLimit
MsgBox (strMsg)
```

**ISTDPeakID Property**
Sets or returns the ISTD peak ID of this named peak. This is a member of the NamedPeak object.

**Type**
Read/Write Long (4 byte integer value)

**Example**
This example displays a message box with the ISTD peak ID for the named peak:

```
strMsg = "The ISTD peak ID for the current " 
```
LimitOfDetection Property
Sets or returns the limit of detection value of this named peak. This is a member of the NamedPeak object.

Type
Read/Write Double (8 byte floating point value)

Example
This example displays a message box with the limit of detection value for the named peak:

```vbscript
strMsg = "The limit of detection value for the current " & "named peak is " & NamedPeak.LimitOfDetection
MsgBox (strMsg)
```

LimitOfQuantitation Property
Sets or returns the limit of quantitation value of this named peak. This is a member of the NamedPeak object.

Type
Read/Write Double (8 byte floating point value)

Example
This example displays a message box with the limit of quantitation value for the named peak:

```vbscript
strMsg = "The limit of quantitation value for the " & "current named peak is " & NamedPeak.LimitOfQuantitation
MsgBox (strMsg)
```

LowConc Property
Sets or returns the low concentration value of this named peak. This is a member of the NamedPeak object.

Type
Read/Write Double (8 byte floating point value)

Example
This example displays a message box with the low concentration value for the named peak:

```vbscript
strMsg = "The low concentration value for the current " & "named peak is " & NamedPeak.LowConc
MsgBox (strMsg)
```
LowSpikeLimit Property

Sets or returns the low spike limit of this named peak. This is a member of the NamedPeak object.

Type
Read/Write Double (8 byte floating point value)

Example
This example displays a message box with the low spike limit for the named peak:

```vbscript
strMsg = "The low spike limit for the current " _
        & "named peak is " & NamedPeak.LowSpikeLimit
MsgBox (strMsg)
```

Margin Property

Sets or returns the margin value of this named peak. This is a member of the NamedPeak object.

Type
Read/Write Double (8 byte floating point value)

Example
This example displays a message box with the margin value for the named peak:

```vbscript
strMsg = "The margin value for the current " _
        & "named peak is " & NamedPeak.Margin
MsgBox (strMsg)
```

MaxLevelUsed Property

Returns the highest level used of this named peak. This is a member of the NamedPeak object.

Type
Read Only Long (4 byte integer value)

Example
This example displays a message box with the max level used for the named peak:

```vbscript
strMsg = "The max level used for the current " _
        & "named peak is " & NamedPeak.MaxLevelUsed
MsgBox (strMsg)
```

PeakEnabled Property

Sets or returns the peak-enabled flag of this named peak. This is a member of the NamedPeak object.
Type
Read/Write Boolean

Example
This example displays a message box with the status of the peak-enabled flag for the named peak:

```vba
strMsg = "The peak-enabled flag for the current " & "named peak is set to " & NamedPeak.PEakEnabled
MsgBox (strMsg)
```

**PeakID Property**

Returns the peak ID of this named peak. This is a member of the NamedPeak object.

Type
Read Only Long (4 byte integer value)

Example
This example displays a message box with the peak ID for the named peak:

```vba
strMsg = "The peak ID for the current " & "named peak is " & NamedPeak.PeakID
MsgBox (strMsg)
```

**PeakName Property**

Sets or returns the peak name of this named peak. This is a member of the NamedPeak object.

Type
Read/Write String

Example
This example displays a message box with the peak name for the named peak:

```vba
strMsg = "The peak name for the current " & "named peak is " & NamedPeak.PeakName
MsgBox (strMsg)
```

**Quantization Property**

Sets or returns the quantization type of this named peak. This is a member of the NamedPeak object.

The quantization type is one of the following constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Area</td>
</tr>
<tr>
<td>1</td>
<td>Height</td>
</tr>
</tbody>
</table>
**Type**  
Read/Write Long (4 byte integer value)

**Example**  
This example sets the quantization type to be Area:

```vbnet
NamedPeak.Quantization = 0
```

---

**RefPeakID Property**

Sets or returns the reference peak ID of this named peak. This is a member of the NamedPeak object.

**Type**  
Read/Write Long (4 byte integer value)

**Example**  
This example displays a message box with the reference peak ID for the named peak:

```vbnet
strMsg = "The reference peak ID for the current "  _
   & "named peak is " & NamedPeak.RefPeakID
MsgBox (strMsg)
```

---

**RemoveLevel Method**

Removes (sets to blank) the value for the given level number of this named peak, thereby disabling this level. This is a member of the NamedPeak object.

**Returns**  
Void

**Syntax**

```vbnet
RemoveLevel (LevelNumber)
```

- **LevelNumber**: Long (4 byte integer value). This is the level number that the user wishes to set the value for to blank, disabling the use of this level.

**Example**  
This example disables and sets the value to blank for level 3 of the named peak:

```vbnet
NamedPeak.RemoveLevel(3)
```

---

**RemoveManualRF Method**

Removes (sets to blank) the Manual RF value of this named peak and disables it. This is a member of the NamedPeak object.
Returns
Void

Syntax
RemoveManualRF()

Example
This example disables and sets the value to blank for Manual RF of the named peak:

NamedPeak.RemoveManualRF

ResolutionPeakID Property
Sets or returns the resolution peak ID of this named peak. This is a member of the NamedPeak object.

Type
Read/Write Long (4 byte integer value)

Example
This example displays a message box with the resolution peak ID for the named peak:

strMsg = "The resolution peak ID for the current " & "named peak is " & NamedPeak.ResolutionPeakID
MsgBox (strMsg)

RF_RSDLimit Property
Sets or returns the RF RSD limit of this named peak. This is a member of the NamedPeak object.

Type
Read/Write Double (8 byte floating point value)

Example
This example displays a message box with the RF RSD limit for the named peak:

strMsg = "The RF RSD limit for the current " & "named peak is " & NamedPeak.RF_RSDLimit
MsgBox (strMsg)

RT_UpdateListIndex Property
Sets or returns the RT Update List Index type of this named peak. This is a member of the NamedPeak object.
The index type is one of the following constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None</td>
</tr>
<tr>
<td>1</td>
<td>Run</td>
</tr>
<tr>
<td>2</td>
<td>Calib</td>
</tr>
<tr>
<td>3</td>
<td>Run &amp; Calib</td>
</tr>
</tbody>
</table>

**Type**  
Read/Write Long (4 byte integer value)

**Example**  
This example sets the RT Update List Index type to be Run & Calib:

```plaintext
NamedPeak.RT_UpdateListIndex = 3
```

**ScaleFactor Property**  
Sets or returns the scale factor of this named peak. This is a member of the `NamedPeak` object.

The scale factor is one of the following constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None</td>
</tr>
<tr>
<td>1</td>
<td>1/x</td>
</tr>
<tr>
<td>2</td>
<td>1/x^2</td>
</tr>
<tr>
<td>3</td>
<td>ln(x)</td>
</tr>
<tr>
<td>4</td>
<td>1/ln(x)</td>
</tr>
<tr>
<td>5</td>
<td>sqrt(x)</td>
</tr>
<tr>
<td>6</td>
<td>x^2</td>
</tr>
</tbody>
</table>

**Type**  
Read/Write Long (4 byte integer value)

**Example**  
This example sets the scale factor to be `sqrt(x)`:  

```plaintext
NamedPeak.ScaleFactor = 5
```

**SetLevelData Method**  
Sets the level value for the given level for the named peak. This is a member of the `NamedPeak` object.

**Returns**  
Void

**Syntax**  
`SetLevelData(LevelNumber, LevelValue)`

**LevelNumber**  
Long (4 byte integer value). This is the level number that the user wishes to set a value for.
**LevelValue**  
Single (4 byte floating point value). This is the new value to set for the given LevelNumber.

**Example**  
This example sets the level #2 value to 97.0 for the named peak:

```csharp
Call NamedPeak.SetLevelData(2, 97#)
```

**SetManualRF Method**  
Sets and enables the Manual RF value for the named peak. This is a member of the `NamedPeak` object.

**Returns**  
Void

**Syntax**  
`SetManualRF(ManualRFValue)`

**ManualRFValue**  
Double (8 byte floating point value). This is the new Manual RF value for the named peak.

**Example**  
This example enables and sets the Manual RF value to 675.0 for the named peak:

```csharp
Call NamedPeak.SetManualRF(675#)
```

**SetNamedPeakData Method**  
Sends all the Named Peak values to the open method. This is a member of the `NamedPeak` object.

**Returns**  
Void

**Syntax**  
`SetNamedPeakData()`

**Example**  
This example sends all the Named Peak properties to the open method:

```csharp
NamedPeak.SetNamedPeakData
```

**Remarks**  
The following properties will be sent to the method using this call:

- CalibFlag
- CalibWeight
- CheckStdConc1
- CheckStdConc2
- CheckStdConc3
- CheckStdConc4
- CheckStdConc5
Any properties not set through automation will retain their current value.

**SimThreshold Property**

Sets or returns the similarity threshold of this named peak. This is a member of the [NamedPeak](#) object.

**Type**
Read/Write Double (8 byte floating point value)

**Example**
This example displays a message box with the similarity threshold for the named peak:

```python
strMsg = "The similarity threshold for the current " _
```
& "named peak is " & NamedPeak.SimThreshold
MsgBox (strMsg)

**SpectrumFileName Property**
Sets or returns the spectrum file name of this named peak. This is a member of the **NamedPeak** object.

**Type**
Read/Write String

**Example**
This example displays a message box with the spectrum file name for the named peak:

```
strMsg = "The spectrum file name for the current "
        & "named peak is " & NamedPeak.SpectrumFileName
MsgBox (strMsg)
```

**Spike1Amount Property**
Sets or returns the spike1 amount of this named peak. This is a member of the **NamedPeak** object.

**Type**
Read/Write Double (8 byte floating point value)

**Example**
This example displays a message box with the spike1 amount for the named peak:

```
strMsg = "The spike1 amount for the current "
        & "named peak is " & NamedPeak.Spike1Amount
MsgBox (strMsg)
```

**Spike2Amount Property**
Sets or returns the spike2 amount of this named peak. This is a member of the **NamedPeak** object.

**Type**
Read/Write Double (8 byte floating point value)

**Example**
This example displays a message box with the spike2 amount for the named peak:

```
strMsg = "The spike2 amount for the current "
        & "named peak is " & NamedPeak.Spike2Amount
MsgBox (strMsg)
```
**StdID Property**

Sets or returns the standard ID of this named peak. This is a member of the `NamedPeak` object.

**Type**
Read/Write Long (4 byte integer value)

**Example**
This example displays a message box with the standard ID for the named peak:

```vba
strMsg = "The standard ID for the current " & "named peak is " & NamedPeak.StdID
MsgBox (strMsg)
```

**StdMult Property**

Sets or returns the standard multiplier of this named peak. This is a member of the `NamedPeak` object.

**Type**
Read/Write Double (8 byte floating point value)

**Example**
This example displays a message box with the standard multiplier for the named peak:

```vba
strMsg = "The standard multiplier for the current " & "named peak is " & NamedPeak.StdMult
MsgBox (strMsg)
```

**WeightingMethod Property**

Sets or returns the weighting method of this named peak. This is a member of the `NamedPeak` object.

The weighting method is one of the following constants:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None</td>
</tr>
<tr>
<td>1</td>
<td>1/Response</td>
</tr>
<tr>
<td>2</td>
<td>1/Response^2</td>
</tr>
<tr>
<td>3</td>
<td>1/Amount</td>
</tr>
<tr>
<td>4</td>
<td>1/Amount^2</td>
</tr>
</tbody>
</table>

**Type**
Read/Write Long (4 byte integer value)

**Example**
This example sets the weighting method to be 1/Response:

```vba
NamedPeak.WeightingMethod = 1
```
**ESignatures Object**

This object contains a collection of all the ESignature objects that are currently available. The DataFileInfo object returns a pointer to this object. Applications can enumerate through the collection of e-signatures by using For … Next statements or by using For … Each statements. For an overview of all objects, see the Object Hierarchy.

**Example**

This example displays the user name of each e-signature in the collection using For … Each statements:

```
For Each ESignature In ESignatures
    MsgBox ("User Name: " & ESignature.UserName)
Next
```

This example displays the name of each named peak in the collection using For … Next statements:

```
nCount = ESignatures.Count
For nIndex = 0 To nCount - 1
    MsgBox ("User Name: " & ESignatures(nIndex).UserName)
Next nIndex
```

**Count Property**

Returns the number of e-signatures. This is a member of the ESignatures object.

**Type**

Read Only Long (4 byte integer value)

**Example**

This example gets the number of users who signed a data doc:

```
Dim nTotalSignatures As Integer
nTotalSignatures = DataFileInfo.ESignatures.Count
```

**Item Property**

Returns the ESignature object at the given indexed location. This is a member of the ESignatures object.

**Type**

Read Only Object
Item(Index)

**Index**
Required Long (4 byte integer value). This is a zero based index that specifies the ESignature object to be returned.

**Example**
This example displays the user name of the e-signatures to the user:

```vbscript
nCount = ESignatures.Count
For nIndex = 0 To nCount - 1
    MsgBox ("User Name: " & ESignatures(nIndex).UserName)
Next nIndex
```

**Remarks**
Item is the default member of this collection, so the following two lines of code are equivalent:

```vbscript
ESignatures.Item(1)
ESignatures(1)
```

**Name Property**

Returns the name of this object. This is a member of the ESignatures object.

**Type**
Read Only String

**Example**
This example displays a message box with the name of this object to the user:

```vbscript
strMsg = "The name of the ESignatures object is " & ESignatures.Name
MsgBox (strMsg)
```
**ESignature Object**

This object represents an e-signature record. The **ESignatures** collection object contains all of the e-signature records that are currently available. For an overview of all objects, see the Object Hierarchy.

**Comment Property**

Returns the comment for this e-signature. This is a member of the **ESignature** object.

**Type**
Read Only String

**Example**
This example gets the comment of the first e-signature:

```vbnet
Dim strComment As String
strComment = Instrument.DataFileInfo.ESignatures(0).Comment
```

**DateTimeStamp Property**

Returns the date and time this e-signature was recorded. This is a member of the **ESignature** object.

**Type**
Read Only String

**Example**
This example gets the date and time the first e-signature was recorded:

```vbnet
Dim strDate As String
strDate = Instrument.DataFileInfo.ESignatures(0).DateTimeStamp
```

**Reason Property**

Returns the reason for this e-signature. This is a member of the **ESignature** object.

**Type**
Read Only String

**Example**
This example gets the reason of the first e-signature:
Dim strReason As String
strReason = Instrument.DataFileInfo.ESignatures(0).Reason

**Role Property**

Returns the role for this e-signature. This is a member of the ESignature object.

**Type**
Read Only String

**Example**
This example gets the role of the first e-signature:

Dim strRole As String
strRole = Instrument.DataFileInfo.ESignatures(0).Role

**UserName Property**

Returns the user name for this e-signature. This is a member of the ESignature object.

**Type**
Read Only String

**Example**
This example gets the user name of the first e-signature:

Dim strUser As String
strUser = Instrument.DataFileInfo.ESignatures(0).UserName