



CmtNA IVI.Net Driver Reference

Revision 21.2 13.08.2021

U.S.: +1.317.222.5400
Latin America: +1.9154.706.5920

Singapore: +65.63.23.6546
EMAE: +44 75 03 69 21 13

Contents

Getting Started	26
CMT VNA IVI.NET Driver	28
Instantiating the IVI.NET Driver	29
Accessing Repeated Capabilities	34
Using Simulation	37
Programming with the IVI.NET driver	38
Using Visual C#	39
CMT.Instruments	50
class NetworkAnalyzer	51
Methods	52
AbortPrint	75
ApplyCalibration	76
ApplySimplifiedCalibration	78
CalKitLoadFromFile	80
CalKitRestore	82
CalKitSaveToFile	83
CalStandardInsert	85
CalStandardRemove	87
CancelCalibration	89
ChannelFrequencyData	91
ClearAverage	93
ClearError	95
ClearRegisterStates	96
ClearRippleLimitTable	98
Close	100
ConfigureFrequencyCenterSpan	101
ConfigureFrequencyStartStop	103
ConfigureGatingCenterSpan	105

Contents

ConfigureGatingStartStop	108
ConfigurePowerCenterSpan	111
ConfigurePowerStartStop	113
ConfigureSweep	115
ConfigureTimeDomainCenterSpan	117
ConfigureTimeDomainStartStop	119
ContinuousAllChannels	121
Disable	123
DisplaySetDefaults	124
ErrorMessage	125
ErrorQuery	127
ExecuteAutoCalCalibration	129
ExecuteAutoCalConfidenceCheck	132
ExecuteAutoCalNPortCalibration	134
ExecuteAutoOrientation	136
ExportLossTable	137
Finalize	139
FunctionExecute	140
GetAttributeViBoolean	142
GetAttributeViInt32	144
GetAttributeViReal64	146
GetAttributeViString	148
GetCalculatedFunctionData	150
GetCalibrationInfo	152
GetCalKitOrderLoad	154
GetCalKitOrderOpen	156
GetCalKitOrderShort	158
GetCalKitOrderThru	160

Contents

GetCalKitOrderTrlLine	162
GetCalKitOrderTrlReflect	164
GetCalKitOrderTrlThru	166
GetCalStandardS1PData	168
GetCalStandardS2PData	171
GetError	175
GetLimitTestData	177
GetLimitTestReportAll	180
GetLimitTestReport	183
GetLimitTestStatus	185
GetMarkerValue	187
GetMaxChannelCount	189
GetMaxTraceCount	191
GetPowerCalibrationTable	193
GetPowerLossCompensationTable	195
GetRippleLimitData	197
GetRippleLimitTestReport	200
GetRippleLimitTestStatus	203
GetSegmentData	205
GetSubclassStdOrder	210
GetTouchstoneFileType	212
HoldAllChannels	215
ImportLossTable	217
InitNPortUserCal	219
InitUserCal	221
LoadTouchstoneFile	223
LockSession	225
MarkerFunctionExecute	227

Contents

MarkerFunctionsMarkerCenter	229
MarkerFunctionsMarkerDelay	231
MarkerFunctionsMarkerRefValue	233
MarkerFunctionsMarkerStart	235
MarkerFunctionsMarkerStop	237
MarkerSetLimitLineResponseOffset	239
MeasurementAutoRefValue	241
MeasurementAutoScale	243
MeasurementDataToMemory	245
MeasurementFetchComplex	247
MeasurementFetchFormatted	250
MeasurementFetchMemoryComplex	253
MeasurementFetchMemoryFormatted	256
MeasurementFetchX	259
MeasurementGetParameter	261
MeasurementSetParameter	263
MeasurePortExtensionOpen	265
MeasurePortExtensionShort	267
NetworkAnalyzer(String, Boolean, Boolean)	269
NetworkAnalyzer(String, Boolean, Boolean, String)	271
PerformCalibrationAction	274
PerformUnkThru2PortCal	277
PowerSensorZeroing	279
PrintOut	281
QueryMarkerMathBandwidth	282
QueryMarkerMathFlatness	285
QueryMarkerMathStatistics	288
QueryMarkerTableData	291

Contents

RecallChannelFromRegister	294
RecallFromFile	296
RecallSegmentTable	298
Reset	300
ResetCalibration	301
ResetWithDefaults	303
RestoreLimitTable	304
RestoreRippleLimitTable	306
RevisionQuery	308
SaveChannelToRegister	310
SaveImage	312
SaveLimitTable	314
SaveRippleLimitTable	316
SaveSegmentTable	318
SaveStateToFile	320
SaveTouchstoneFile	322
SaveTraceData	324
SelfTest	326
SetAttributeViBoolean	328
SetAttributeViInt32	330
SetAttributeViReal64	332
SetAttributeViString	334
SetCalKitOrderLoad	336
SetCalKitOrderOpen	338
SetCalKitOrderShort	340
SetCalKitOrderThru	342
SetCalKitOrderTrlLine	344
SetCalKitOrderTrlReflect	346

Contents

SetCalKitOrderTriThru	348
SetCalStandardS1PData	350
SetCalStandardS2PData	352
SetLimitTestData	356
SetPowerCalibrationTable	359
SetPowerLossCompensationTable	361
SetRippleLimitData	363
SetSegmentData	366
SetSubclassStdOrder	370
SetTouchstoneFileType	372
SystemHide	374
SystemPreset	375
SystemShow	376
TakePowerCalSweep	377
TestBeepComplete	379
TestBeepWarning	380
TimeDomainSetFrequencyLowPass	381
TraceHoldRestart	383
TriggerImmediate	385
TriggerInit	387
TriggerRestart	389
TriggerSingleAndWait	391
TriggerSingle	393
UnlockSession	395
WaitForOperationComplete	397
WaitForSweepComplete	399
WaitForTriggerState	401
class Attributes	403

Contents

AdapterRemovalCutoffFreq	543
AdapterRemovalDelay	544
AdapterRemovalLength	546
AdapterRemovalMedia	548
AdapterRemovalPermittivity	550
AdapterRemovalPort	552
AdapterRemovalUnit	553
AnalysisConversion	555
AnalysisConversionFunction	556
AnalysisDeembedding	558
AnalysisDeembeddingPort	559
AnalysisDeembeddingPortFile	560
AnalysisEmbedding	561
AnalysisEmbeddingPort	562
AnalysisEmbeddingPortFile	563
AnalysisFixtureSimulator	564
AnalysisLimitLineDisplay	565
AnalysisLimitTest	566
AnalysisLimitTestFailSign	567
AnalysisLimitTestPoints	568
AnalysisPortZConversion	569
AnalysisPortZConversionImag	570
AnalysisPortZConversionReal	571
AnalysisPortZConversionZ0	572
AnalysisRippleLimitDisplay	573
AnalysisRippleLimitFailSign	574
AnalysisRippleTest	575
AnalysisRippleValueType	576

Contents

AnalysisTimeDomain	578
AnalysisTimeDomainReflectionType	579
AnalysisTimeDomainUnits	581
AttrPrintColor	583
AutocalAutoOrientation	585
AutocalCharacterization	586
AutocalModuleReady	588
AutocalOrientationPort	589
AutocalTemperature	591
AutocalUnknownThru	592
AutoPortExtensionAdjustMismatch	593
AutoPortExtensionIncludeLoss	594
AutoPortExtensionMethod	595
AutoPortExtensionUserSpanStart	597
AutoPortExtensionUserSpanStop	598
BeepCompleteOn	599
BeepWarning	600
CalibrationAutoSelectZ0	601
CalibrationCorrection	602
CalibrationPortZ0	603
CalibrationTriggerSource	604
CalibrationType	606
CalkitDescription	608
CalkitLabel	609
CalkitSelected	610
CalkitStandardArbitrary	611
CalkitStandardC0	612
CalkitStandardC1	613

Contents

CalkitStandardC2	614
CalkitStandardC3	615
CalkitStandardL0	616
CalkitStandardL1	617
CalkitStandardL2	618
CalkitStandardL3	619
CalkitStandardLabel	620
CalkitStandardMaxFrequency	621
CalkitStandardMinFrequency	622
CalkitStandardOffsetDelay	623
CalkitStandardOffsetLoss	624
CalkitStandardOffsetZ0	625
CalkitStandardType	626
CorrectionState	628
CorrectionStatus	629
CorrectionType	630
DevicePxiChassis	631
DevicePxiSlot	632
DeviceReady	633
DeviceSerialNumber	634
DeviceTemperature	635
DisplayActiveChannel	636
DisplayActiveTrace	637
DisplayBackgroundColor	638
DisplayChannelAllocation	639
DisplayCycleTimeValue	640
DisplayDataTraceColor	641
DisplayGridColor	642

Contents

DisplayInvertColor	643
DisplayMaximizeChannel	644
DisplayMaximizeTrace	645
DisplayMemoryTraceColor	646
DisplaySystemDate	647
DisplaySystemTime	648
DisplayTitleData	649
DisplayTitleLabel	650
DisplayTraceAllocation	651
DisplayTraceCount	652
DisplayTraceDataMath	653
DisplayTraceHoldType	655
DisplayTraceType	657
DisplayUpdate	659
ExternalReferenceRoute	660
ExternalTriggerRoute	662
FrequencyDivider	664
FrequencyMultiplier	665
FrequencyOffset	666
FrequencyOffsetStart	667
FrequencyOffsetStop	668
FrequencyOffsetType	669
FrequencyReceiverDivider	671
FrequencyReceiverMultiplier	672
FrequencyReceiverOffset	673
FrequencyReceiverOffsetStart	674
FrequencyReceiverOffsetStop	675
FrequencySourceDivider	676

Contents

FrequencySourceMultiplier	677
FrequencySourceOffset	678
FrequencySourceOffsetStart	679
FrequencySourceOffsetStop	680
FunctionDomainCoupling	681
FunctionDomainStart	683
FunctionDomainState	684
FunctionDomainStop	685
FunctionPeakExcursion	686
FunctionPeakPolarity	687
FunctionPoints	689
FunctionTargetLevel	691
FunctionTransitionType	692
FunctionType	694
LimitLineResponseOffset	696
LimitLineStimulusOffset	697
LogicalName	698
MarkerMathBandwidthSearch	699
MarkerMathBandwidthSearchRef	700
MarkerMathBandwidthSearchType	701
MarkerMathBandwidthSearchValue	703
MarkerMathFlatness	704
MarkerMathFlatnessStart	705
MarkerMathFlatnessStop	706
MarkerMathStatisticRange	707
MarkerMathStatistics	709
MarkerMathStatisticStart	710
MarkerMathStatisticStop	711

Contents

MarkerPropertiesAlign	712
MarkerPropertiesDataXPosition	713
MarkerPropertiesDataYPosition	714
MarkerPropertiesDiscrete	715
MarkerPropertiesMarkerActiveOnly	716
MarkerPropertiesMarkerCouple	717
MarkerPropertiesMarkerTable	719
MarkersCount	720
MarkerSearchCouple	721
MarkerSearchPeakExcursion	723
MarkerSearchPeakPolarity	724
MarkerSearchRange	726
MarkerSearchStart	727
MarkerSearchStop	728
MarkerSearchTargetTransition	729
MarkerSearchTargetValue	731
MarkerSearchTracking	733
MarkerSearchType	734
MarkerStimulus	736
MeasurementAveraging	737
MeasurementAveragingFactor	738
MeasurementAvgTrigger	739
MeasurementDivisions	741
MeasurementElecDelay	742
MeasurementFormat	743
MeasurementPhaseOffset	745
MeasurementRefPosition	746
MeasurementRefValue	747

Contents

MeasurementScaleDiv	749
MeasurementSmoothing	751
MeasurementSmoothingAperture	752
NumberOfPorts	753
OutputTriggerRoute	754
PortExtensions	756
PortExtensionsFreq1Value	757
PortExtensionsFreq2Value	758
PortExtensionsLdcValue	759
PortExtensionsLoss1State	760
PortExtensionsLoss1Value	761
PortExtensionsLoss2State	762
PortExtensionsLoss2Value	763
PortExtensionsTime	764
PowerCalibrationCorrection	765
PowerCalibrationLossCompensation	766
PowerSensorReady	767
PowerSensorType	768
PowerTripAtOverload	770
PrintDateAndTime	771
PrintInvertImage	772
ReferenceFrequencySource	773
ReferenceMarker	775
SaveTouchstoneFileColumnSeparator	777
SaveTouchstoneFileFormat	779
SaveType	781
SelectedMarker	783
StimulusExtTriggerDelay	785

Contents

StimulusExtTriggerEvent	786
StimulusExtTriggerPolarity	788
StimulusExtTriggerPosition	790
StimulusFrequencyCenter	792
StimulusFrequencyOffset	793
StimulusFrequencySpan	794
StimulusFrequencyStart	795
StimulusFrequencyStop	796
StimulusIfBandwidth	797
StimulusMaxFrequency	799
StimulusMaxPoints	800
StimulusMinFrequency	801
StimulusOutputPower	802
StimulusOutputPowerPort	803
StimulusOutputPowerPortCouple	804
StimulusOutputPowerRfout	805
StimulusOutputPowerSlope	806
StimulusOutputPowerSlopeState	807
StimulusPoints	808
StimulusPowerCenter	809
StimulusPowerCwFrequency	810
StimulusPowerSpan	811
StimulusPowerStart	812
StimulusPowerStop	813
StimulusSegmentDisplayOrder	814
StimulusSweepMeasureDelay	816
StimulusSweepType	817
StimulusTriggerMode	819

Contents

StimulusTriggerOutput	821
StimulusTriggerOutputFunction	822
StimulusTriggerOutputPolarity	824
StimulusTriggerScope	826
StimulusTriggerSource	828
StimulusTriggerState	830
SystemCorrection	832
SystemReadWriteLock	833
SystemTimeoutMilliseconds	834
ThruAdditionCutoffFreq	835
ThruAdditionDelay	836
ThruAdditionLength	838
ThruAdditionMedia	840
ThruAdditionPermittivity	842
ThruAdditionUnit	843
TimeDomainCableCorrection	845
TimeDomainCableFrequency	846
TimeDomainCableLoss	847
TimeDomainCableVelocityFactor	848
TimeDomainCenter	849
TimeDomainGating	850
TimeDomainGatingCenter	851
TimeDomainGatingShape	852
TimeDomainGatingSpan	854
TimeDomainGatingStart	855
TimeDomainGatingStop	856
TimeDomainGatingType	857
TimeDomainImpulseWidth	859

Contents

TimeDomainKaiserBeta	861
TimeDomainSpan	862
TimeDomainStart	863
TimeDomainStop	864
TimeDomainTransformType	866
TimeDomainWindowShape	868
VerificationInterval	870
VerificationLastDate	872
VerificationNextDate	873
class Constants	874
AdapterRemovalMediaCoaxial	894
AdapterRemovalMediaWaveguide	895
AdapterRemovalUnitMeters	896
AdapterRemovalUnitSeconds	897
AnalysisLimitLineMax	898
AnalysisLimitLineMin	899
AnalysisLimitLineOff	900
AnalysisLimitLineSingle	901
AnalysisRippleLimitOff	902
AnalysisRippleLimitOn	903
AnalysisRippleValueAbsolute	904
AnalysisRippleValueMargin	905
AnalysisRippleValueOff	906
Autocal1port	907
Autocal2port	908
Autocal3port	909
Autocal4port	910
AutocalCharacterizationFactory	911

Contents

AutocalCharacterizationUser1	912
AutocalCharacterizationUser2	913
AutocalCharacterizationUser3	914
AutocalOnePath2port	915
AutocalOrientationPortA	916
AutocalOrientationPortB	917
AutocalOrientationPortC	918
AutocalOrientationPortD	919
AutoPortExtensionActiveMarker	920
AutoPortExtensionCurrentSpan	921
AutoPortExtensionUserSpan	922
CalibrationActionIsolation	923
CalibrationActionLoad	924
CalibrationActionOpen	925
CalibrationActionPowerCal	926
CalibrationActionReferenceReceiverCal	927
CalibrationActionShort	928
CalibrationActionTestReceiverCal	929
CalibrationActionThru	930
CalibrationTriggerSourceInternal	931
CalibrationTriggerSourceSystem	932
CalibrationType1path2port	933
CalibrationType1portSol	934
CalibrationType2portSolt	935
CalibrationType2portTrl	936
CalibrationType3portSolt	937
CalibrationType3portTrl	938
CalibrationType4portSolt	939

Contents

CalibrationType4portTrl	940
CalibrationTypeAdapterRemoval	941
CalibrationTypeNotDefined	942
CalibrationTypeResponseOpen	943
CalibrationTypeResponseShort	944
CalibrationTypeResponseThru	945
CalkitStandardDataBased	946
CalkitStandardLoad	947
CalkitStandardNone	948
CalkitStandardOpen	949
CalkitStandardShort	950
CalkitStandardSlidingLoad	951
CalkitStandardThruDelay	952
CalkitStandardUnknThru	953
ChannelStateRegisterA	954
ChannelStateRegisterB	955
ChannelStateRegisterC	956
ChannelStateRegisterD	957
ConversionFunctionConjugation	958
ConversionFunctionSInverse	959
ConversionFunctionYReflection	960
ConversionFunctionYTransmission	961
ConversionFunctionYTransShunt	962
ConversionFunctionZReflection	963
ConversionFunctionZTransmission	964
ConversionFunctionZTransShunt	965
CorrectionStateCorrection	966
CorrectionStateExtrapolation	967

Contents

CorrectionStateInterpolation	968
CorrectionStateNone	969
DisplayTraceData	970
DisplayTraceDataAndMemory	971
DisplayTraceMemory	972
DisplayTraceOff	973
ExtTriggerEventOnPoint	974
ExtTriggerEventOnSweep	975
ExtTriggerPolarityNegative	976
ExtTriggerPolarityPositive	977
ExtTriggerPositionBsampling	978
ExtTriggerPositionBsetup	979
FrequencyOffsetTypePortPort	980
FrequencyOffsetTypeSourceReceiver	981
FunctionAllPeaks	982
FunctionAllTarget	983
FunctionMaximum	984
FunctionMean	985
FunctionMinimum	986
FunctionPeak	987
FunctionPeakPolarityBoth	988
FunctionPeakPolarityNegative	989
FunctionPeakPolarityPositive	990
FunctionPeakToPeak	991
FunctionStandardDeviation	992
FunctionTransitionTypeBoth	993
FunctionTransitionTypeNegative	994
FunctionTransitionTypePositive	995

Contents

MarkerMathBandwidthSearchBandpass	996
MarkerMathBandwidthSearchNotch	997
MarkerMathBandwidthSearchRefMarker	998
MarkerMathBandwidthSearchRefMinimum	999
MarkerMathBandwidthSearchRefMaximum	1000
MarkerPropertiesAlignHorizontal	1001
MarkerPropertiesAlignOff	1002
MarkerPropertiesAlignVertical	1003
MarkerSearchPeakPolarityBoth	1004
MarkerSearchPeakPolarityNegative	1005
MarkerSearchPeakPolarityPositive	1006
MarkerSearchTargetTransitionBoth	1007
MarkerSearchTargetTransitionNegative	1008
MarkerSearchTargetTransitionPositive	1009
MarkerSearchTypeMaximum	1010
MarkerSearchTypeMinimum	1011
MarkerSearchTypePeak	1012
MarkerSearchTypePeakLeft	1013
MarkerSearchTypePeakRight	1014
MarkerSearchTypeTarget	1015
MarkerSearchTypeTargetLeft	1016
MarkerSearchTypeTargetRight	1017
MeasurementFormatExpandPhase	1018
MeasurementFormatGroupDelay	1019
MeasurementFormatImag	1020
MeasurementFormatLinMag	1021
MeasurementFormatLogMag	1022
MeasurementFormatPhase	1023

Contents

MeasurementFormatPolarLin	1024
MeasurementFormatPolarLog	1025
MeasurementFormatPolarReallImage	1026
MeasurementFormatReal	1027
MeasurementFormatSmithAdmittance	1028
MeasurementFormatSmithComplex	1029
MeasurementFormatSmithLin	1030
MeasurementFormatSmithLog	1031
MeasurementFormatSmithReallImage	1032
MeasurementTypeAbsoluteR	1033
MeasurementFormatSwr	1034
MeasurementTypeAbsoluteT	1035
PowerSensorNrpxt	1036
PowerSensorNrpxz	1037
PowerSensorNrps	1038
PowerSensorU200x	1039
PowerSensorU848x	1040
PrintBlackAndWhite	1041
PrintColor	1042
PrintGrayScale	1043
ReferenceFrequencySourceExternal	1044
ReferenceFrequencySourceInternal	1045
ReferenceRouteFront	1046
ReferenceRouteRear	1047
SaveTouchstoneFileColumnSeparatorSpace	1048
SaveTouchstoneFileColumnSeparatorTab	1049
SaveTouchstoneFileFormatDbAngle	1050
SaveTouchstoneFileFormatMagnitudeAngle	1051

Contents

SaveTouchstoneFileFormatReallImage	1052
SaveTouchstoneFileS1p	1053
SaveTouchstoneFileS2p	1054
SaveTouchstoneFileS3p	1055
SaveTouchstoneFileS4p	1056
SaveTypeAll	1057
SaveTypeState	1058
SaveTypeStateAndCal	1059
SaveTypeStateAndCalAndMem	1060
SaveTypeStateAndTrace	1061
SegmentFrequencyOrder	1062
SegmentIndexOrder	1063
StimulusTriggerStateHold	1064
StimulusTriggerStateMeasure	1065
StimulusTriggerStateWait	1066
SweepTypeLinFrequency	1067
SweepTypeLogFrequency	1068
SweepTypePower	1069
SweepTypeSegment	1070
ThruAdditionMediaCoaxial	1071
ThruAdditionMediaWaveguide	1072
ThruAdditionUnitMeters	1073
ThruAdditionUnitSeconds	1074
TimeDomainGatingBandpass	1075
TimeDomainGatingNotch	1076
TimeDomainGatingShapeMaximum	1077
TimeDomainGatingShapeMinimum	1078
TimeDomainGatingShapeNormal	1079

Contents

TimeDomainGatingShapeWide	1080
TimeDomainReflectionOneWay	1081
TimeDomainReflectionRoundTrip	1082
TimeDomainTransformTypeBandpass	1083
TimeDomainTransformTypeLowpassImpulse	1084
TimeDomainTransformTypeLowpassStep	1085
TimeDomainUnitsFeet	1086
TimeDomainUnitsMeters	1087
TimeDomainUnitsSeconds	1088
TimeDomainWindowShapeArbitrary	1089
TimeDomainWindowShapeMaximum	1090
TimeDomainWindowShapeMinimum	1091
TimeDomainWindowShapeNormal	1092
TraceDataMathAdd	1093
TraceDataMathDiv	1094
TraceDataMathMult	1095
TraceDataMathOff	1096
TraceDataMathSubt	1097
TraceHoldMax	1098
TraceHoldMin	1099
TraceHoldOff	1100
TriggerModeContinuous	1101
TriggerModeHold	1102
TriggerModeSingle	1103
TriggerOutputFunctionAsampling	1104
TriggerOutputFunctionBsampling	1105
TriggerOutputFunctionBsetup	1106
TriggerOutputFunctionMeasurement	1107

Contents

TriggerOutputFunctionReadyForTrigger	1108
TriggerOutputFunctionSweepEnd	1109
TriggerOutputPolarityNegative	1110
TriggerOutputPolarityPositive	1111
TriggerRoutePxiTrig0	1112
TriggerRoutePxiTrig1	1113
TriggerRoutePxiTrig2	1114
TriggerRoutePxiTrig3	1115
TriggerRoutePxiTrig4	1116
TriggerRoutePxiTrig5	1117
TriggerRoutePxiTrig6	1118
TriggerRoutePxiTrig7	1119
TriggerRouteSmb	1120
TriggerRouteStar	1121
TriggerScopeActiveChannel	1122
TriggerScopeAllChannel	1123
TriggerSourceBus	1124
TriggerSourceExternal	1125
TriggerSourceInternal	1126
TriggerSourceManual	1127
UploadTouchstoneFileToActiveTraceMemory	1128
UploadTouchstoneFileToSParameters	1129
Installation	1130
Copyright	1133

Getting Started

Welcome to the CMT VNA IVI.NET driver version **21.2.0**.

The CMT VNA IVI.NET driver simplifies the creation and maintenance of applications in a variety of development environments.

IVI drivers control, via programming, the instrumentation, while providing a greater degree of instrument interchangeability and code reuse.

The CMT VNA IVI.NET driver supports compiling application programs for 32 or 64 bit platform.

Programming with CMT.IVI.Net driver

The CMT VNA IVI.NET driver can be used in the following application development environments: Visual Basic.NET, Visual C#, MATLAB. For a detail discription see [Programming with the IVI.NET driver](#).

CMT.IVI.Net Driver Reference

The CMT VNA IVI.NET driver is represented by a single NetworkAnalyzer interface with the list of methods, lists of attributes and constants. For a detail discription see [CMT.Instruments](#).

IVI Overview

General information on IVI features common to all IVI drivers as well as compliance and installation information.

Compliance information for installation and update CMT VNA IVI.NET driver see in [Installation](#).

Driver Identification

Identification Category	Description
Driver Revision	21.2.0
Driver Vendor	Copper Mountain Technologies
Driver Description	CMT VNA IVI.NET Driver Assembly

Identification Category	Description
Supported Models	PXI-s5090

Required components for compilation

[.NET Framework 4.0](#)

[Visual Studio 2010](#)

VISA library

[MI Shared Components](#) - Usage Guides, Specifications, Shared Components Downloads

[MI.NET Shared Components](#)

IVI-C Driver (part of the software supplied with the device)

Web Sites

[Copper Mountain Technologies](#)

[MSDN Online](#)

CMT VNA IVI.NET Driver

The CMT VNA IVI.NET driver includes a fully compliant IVI.NET driver.

The topics in this section describe how to perform specific programming tasks using the IVI.NET interfaces available with the CMT VNA IVI.NET driver such as:

- Steps required to properly instantiate an instance of the IVI.NET driver (See [Instantiating the IVI.NET Driver](#)).
- Gaining access to the specialized functionality of the CMT VNA IVI.NET driver (See [Accessing Instrument-Specific Functionality](#)).
- Information on using the simulation capabilities of the CMT VNA IVI.NET driver (See [Using Simulation](#)).
- CMT VNA IVI.NET driver usage in a various Application Development Environments (ADEs) (See [Programming with the IVI.NET driver in various development environment](#)).
- Reference documentation for each IVI.NET method and property (See [CMT.Instruments](#)).

Instantiating the IVI.NET Driver

The simplest way of creating an instance of the CMT VNA IVI.NET driver is to call one of the driver's constructors. This gives access to the full capabilities of the driver, and provides the most complete access to the supported capabilities of instrument. For example:

```
using CMT.Instruments;  
  
NetworkAnalyzer nwa = new NetworkAnalyzer(resourceName, idQuery,  
reset, optionsString);
```

The details of creating a driver instance using this way are described below.

Create an instance of the driver by calling one of the driver's constructors, unless you are specifically writing an interchangeable application. This way provides full access to the driver's functionality.

The IVI.NET Driver Constructor

The constructor for an IVI.NET driver function offers a variety of options and parameters that control fundamental aspects of the driver's behavior. Understand what options are available and how they impact driver operation. For details on constructor syntax and overloads, see [NetworkAnalyzer Constructor](#).

ResourceName

The resource name parameter can be a logical name present in the VI Configuration Store or a physical resource descriptor. If the ResourceName is a logical name, then additional options stored in the VI Configuration Store will be loaded and processed. This allows client code to reuse driver option settings and to keep the driver code as concise as possible, by removing settings that can be stored in the VI Configuration Store. It is important to note that any parameters or options passed directly to the [NetworkAnalyzer Constructor](#) via the OptionString parameter (discussed below) override any settings found in the VI Configuration Store.

Interface	Physical Resource Descriptor Syntax
PXI	PXI[bus]SLOT[#slot] Example: PXI1SLOT3

IdQuery

If this is enabled, the driver will query the instrument model and compare it with a list of instrument models that are supported by the driver. If the model is not supported,

[NetworkAnalyzer Constructor](#) will generate `IVI.Driver.IOException` with mistake error. Exception handling block allows to get a string with error description.

Reset

If this is enabled, the driver will perform a reset of the instrument. If the reset fails, [NetworkAnalyzer Constructor](#) will generate `IVI.Driver.IOException` with mistake error. Exception handling block allows to get a string with error description.

OptionString

The `OptionString` allows the user to pass optional settings to the driver. Any setting that is not specified has a default value as specified by IVI.

Option Name	Description	Default
QueryInstrStatus	Specifies whether the IVI specific driver queries the instrument status at the end of each user operation. Querying the instrument status is very useful for debugging. After validating the program, the user can set this attribute to False to disable status checking and maximize performance. The user specifies this value for the entire IVI driver session.	false
Simulate	Specifies whether or not the IVI specific driver simulates instrument driver I/O operations. If simulation is enabled, the specific driver functions do not perform instrument I/O. For output parameters that represent instrument data, the specific driver functions return simulated values.	false
DriverSetup	Specifies additional settings supported by the driver, but not defined by IVI.	""

DriverSetup

This is used to specify settings that are supported by the driver but not defined by IVI. If the Options String parameter contains an assignment for the Driver Setup attribute, [NetworkAnalyzer Constructor](#) assumes that everything following 'DriverSetup=' is a part of the assignment. The following settings are supported by the CMT VNA IVI.NET driver.

Option Name	Description	Default
Visible	If true, UI is visible.	false
WaitReady	If true, initialization function waits for the device to be ready (up to 15 seconds).	false
DirectVisaAddresses	<p>If true, its VISA Address is used instead of the ResourceName. For example: VISA Address: "TCPIP0::127.0.0.1::hislip0::INSTR", key with the option line: "DirectVisaAddress=true".</p> <p>DirectVisaAddress allows the user to control Copper Mountain Technologies Vector Network Analyzers (VNA) (S2VNA, PXI S2VNA) that support the HiSLIP protocol and a compatible automation command set. However, there are limitations:</p> <ul style="list-style-type: none"> • Does not run server applications (S2VNA, PXI-VNA). • Not combined with settings that change the state at startup: "Visible = ", "Simulate=". 	false

Example:

The following code initializes the driver in simulation mode.

C#

```
using CMT.Instruments;
using Ivi.Driver;

namespace TestIvi.net
{
    class Program
    {
        static void Main(string[] args)
        {
```

C#

```
string resourceName = "PXI1Slot2";
bool idQuery = false;
bool reset = true;
string optionsString = "Visible=true, Simulate=true";
int Status;
string errorMessage;

NetworkAnalyzer nwa = (NetworkAnalyzer)null;
try
{
    nwa = new NetworkAnalyzer(resourceName, idQuery, reset,
optionsString);
}
catch (Mi.Driver.IOException ex)
{
    Console.WriteLine(ex.ToString());
    Console.ReadKey();
    return;
}

... // Driver application code here...

if (Status != 0)
{
    nwa.ErrorMessage(Status, out errorMessage);
    Console.Write("Error: " + errorMessage);
}
nwa.Close();
}
```


C#

}

Accessing Repeated Capabilities

This section provides information on the supported repeated capabilities and repeated capability names for the CMT VNA IVI.NET driver.

Repeated Capabilities

The following table provides detailed information on each repeated capability supported by the CMT VNA IVI.NET driver.

Repeated Capability	IVI.NET Style	Physical Names		
Channel	String parameter	Channel1	Channel9	
		Channel2	Channel10	
		Channel3	Channel11	
		Channel4	Channel12	
		Channel5	Channel13	
		Channel6	Channel14	
		Channel7	Channel15	
		Channel8	Channel16	
Measurement	String parameter	Measurement1	Measurement9	
		Measurement2	Measurement10	
		Measurement3	Measurement11	
		Measurement4	Measurement12	
		Measurement5	Measurement13	
		Measurement6	Measurement14	
		Measurement7	Measurement15	

Repeated Capability	IVI.NET Style	Physical Names		
		Measurement8	Measurement14 Measurement15 Measurement16	
Marker	String parameter	Marker1 Marker2 Marker3 Marker4 Marker5 Marker6 Marker7 Marker8	Marker9 Marker10 Marker11 Marker12 Marker13 Marker14 Marker15 Marker16	
Standard	String parameter	Standard1 Standard2 Standard3 Standard4 Standard5 Standard6 Standard7	Standard11 Standard12 Standard13 Standard14 Standard15 Standard16 Standard17	Standard21 Standard22 Standard23 Standard24 Standard25 Standard26 Standard27

Repeated Capability	IVI.NET Style	Physical Names		
		Standard8	Standard18	Standard28
		Standard9	Standard19	Standard29
		Standard10	Standard20	Standard30
Port	String parameter	Port1		
		Port2		
		Port3		
		Port4		
		Port5		
		Port6		
		Port7		
		Port8		

Using Simulation

An MI driver can operate without the presence of an actual instrument by using simulation mode. When simulation mode is turned on, the driver performs no instrument communication, but attempts to generate results which allow to reasonably execute client applications.

To enable simulation mode, pass the "simulate=true" option string parameter in the [NetworkAnalyzer Constructor](#).

NOTE

It is impossible to enable simulation mode if the driver is initialized with disabling. However, if an MI.NET driver instance is made with simulation turned on, an exception will be got when trying to disable the simulation. This is because that it is impossible to initiate live I/O session with an instrument after the MI.NET driver constructor is called.

Programming with the IVI.NET driver

IVI.NET drivers are implemented using standard Windows .NET technology. Consequently, IVI.NET drivers can be used in a wide variety of Application Development Environments.

The following topic provides detailed instructions on how to enable IVI.NET driver in a Visual C# project (See [Using Visual C#](#)).

Using Visual C#

This section describes how to use the CMT VNA IVI.NET driver from Visual C#.

Referencing the Driver

In order to access any of the driver interfaces, the proper references to the driver assembly.

The procedure of creating proper references to the driver assembly:

1. In Solution Explorer, right-click on **References** and select **Add Reference**.
2. Open the **Assemblies** folder and select **Extensions**.
3. Select the following assemblies:
 - Ivi.Driver (latest version for target platform)
 - [CMT.Instruments](#)
4. Click the checkbox next to each assembly to select it, and then **OK**.
5. Each of the above assemblies should now appear under the **References** node in solution explorer.

All data types (interfaces and enums) are contained within namespaces. Usually the namespace-qualified name must be used, but the C# using statement allows the type name to be used directly.

C#

```
using CMT.Instruments;  
using Ivi.Driver;
```

Instantiating the Driver

The driver can be instantiated by using the new operator directly on the driver constructor.

Calling the **Constructor** will establish an I/O connection to an instrument (often referred to as an "I/O session") or setup the driver to work in simulation mode.

NOTE

Call **Close** at the end of the program. This is required by the VI specifications, as the driver can behave unpredictably if the program is not closed. Any resources held by the driver may not be properly released if **Close** is not called.

The driver is typically instantiated using a resource descriptor, often a VISA resource descriptor, as shown in the example below.

C#

```
NetworkAnalyzer nwa = new NetworkAnalyzer(resourceName, idQuery, reset,
optionsString);
nwa.Close();
```


Instantiating Using Constructor Options

The way VI-defined options and driver-specific options can be passed to the driver **Constructor** is shown below.

For more details on the **Constructor** and **Options** see [Instantiating the VI.NET Driver](#).

C#

```
namespace TestVi.net
{
    class Program
    {
        static void Main(string[] args)
        {
            string resourceName = "PXI1Slot2";
            bool idQuery = false;
            bool reset = true;
            string optionsString = "Visible=true, Simulate=true";
            int Status;
            string errorMessage;

            NetworkAnalyzer nwa = (NetworkAnalyzer)null;
            try
            {
                nwa = new NetworkAnalyzer(resourceName, idQuery, reset,
optionsString);
            }
            catch (Vi.Driver.IOException ex)
            {
                Console.WriteLine(ex.ToString());
                Console.ReadKey();
                return;
            }
        }
    }
}
```

C#

```
    }

    ... // Driver application code here...

    if (Status != 0)
    {
        nwa.ErrorMessage(Status, out errorMessage);
        Console.WriteLine("Error: " + errorMessage);
    }
    nwa.Close();
}
}
```

Accessing Methods and Properties

Each set of capabilities (inherent, class-compliant or instrument specific) is defined as a hierarchy of interfaces containing methods and properties. A property can return a pointer to another interface -- this is how the hierarchy is built.

NOTE

The code below is for illustration only. It may be from other drivers, and may not apply directly to this driver.

Simple Data Types

C#

```
int Points;
// Get the value of a integer attribute StimulusPoints
nwa.GetAttributeViInt32("Channel1", Attributes.StimulusPoints, out Points);
// Set the value of a integer attribute StimulusPoints
nwa.SetAttributeViInt32("Channel1", Attributes.StimulusPoints, 201);

double FrequencyStart;
// Get the value of a double attribute StimulusFrequencyStart
nwa.GetAttributeViReal64("Channel1", Attributes.StimulusFrequencyStart, out
FrequencyStart);
// Set the value of a double attribute StimulusFrequencyStart
nwa.SetAttributeViReal64("Channel1",    Attributes.StimulusFrequencyStart,
1e8);
```

Strings

C#

```
string sValue = "";
// Get the value of a string attribute CalkitStandardLabel
nwa.GetAttributeViString("Channel1",    Attributes.CalkitStandardLabel,    ref
sValue);
// Set the value of a string attribute CalkitStandardLabel
```

C#

```
nwa.SetAttributeViString("Channel1", Attributes.CalkitStandardLabel, "My Load -F-");
```

Arrays

The example below demonstrates calling a method with an array parameter marked as [out,retval].

C#

```
int Points;
double[] Frequencies = new double[201];
// Read an array of Frequencies values
Status = nwa.ChannelFrequencyData("Channel1", Frequencies, out Points);
// Set arrays for Limit testing
int[] LineTypes = { Constants.AnalysisLimitLineMax,
Constants.AnalysisLimitLineMin, Constants.AnalysisLimitLineMax };
double[] BeginArguments = {1e8, 1e9, 2e9};
double[] EndArguments = {9e8, 1.9e9, 3e9};
double[] BeginValues = {-10, 10, -8};
double[] EndValues = {-8, 10, -10};
nwa.SetLimitTestData("Channel1:Measurement1", LineTypes,
BeginArguments, EndArguments, BeginValues, EndValues);
```

Complete Example

The following code demonstrates a simple console application that instantiates the driver, reads some properties, checks the instrument for errors, and closes the driver.

C#

```
using System;
using System.Collections.Generic;
```

C#

```
using System.Linq;
using System.Text;
using CMT.Instruments;
using Ivi.Driver;

namespace TestIvi.net
{
    class Program
    {
        static void Main(string[] args)
        {
            string resourceName = "PXI1Slot2";
            bool idQuery = false;
            bool reset = true;
            string optionsString = "Visible=true, Simulate=true";

            string sValue = "";
            int iValue;
            int Status;
            string errorMessage;

            double[] Frequencies;
            double[] realData;
            double[] imagData;

            NetworkAnalyzer nwa = (NetworkAnalyzer)null;
            try
            {
                nwa = new NetworkAnalyzer(resourceName, idQuery, reset,
optionsString);
```

```
    }  
    catch (Mi.Driver.IOException ex)  
    {  
        Console.WriteLine(ex.ToString());  
        Console.ReadKey();  
        return;  
    }  
  
        if ((Status = nwa.GetAttributeViString("", Attributes.LogicalName, ref  
sValue)) != 0)  
            goto Error;  
        Console.WriteLine("LogicalName: " + sValue);  
  
                if ((Status = nwa.GetAttributeViString("",  
Attributes.DeviceSerialNumber, ref sValue)) != 0)  
                    goto Error;  
                Console.WriteLine("Serial: " + sValue);  
  
        if ((Status = nwa.GetAttributeViInt32("", Attributes.NumberOfPorts, out  
iValue)) != 0)  
            goto Error;  
        Console.WriteLine("Number of Ports: " + iValue);  
  
        double FrequencyStart = 1e9;  
        double FrequencyStop = 3e9;  
        int Points = 21;  
  
                if ((Status = nwa.SetAttributeViReal64("Channel1",  
Attributes.StimulusFrequencyStart, FrequencyStart)) != 0)  
                    goto Error;
```

C#

```
        if ((Status = nwa.SetAttributeViReal64("Channel1",
Attributes.StimulusFrequencyStop, FrequencyStop)) != 0)
            goto Error;

        if ((Status = nwa.SetAttributeViInt32("Channel1",
Attributes.StimulusPoints, Points)) != 0)
            goto Error;

        if ((Status = nwa.GetAttributeViReal64("Channel1",
Attributes.StimulusFrequencyStart, out FrequencyStart)) != 0)
            goto Error;
        Console.WriteLine("Frequency Start: " + FrequencyStart);

        if ((Status = nwa.GetAttributeViReal64("Channel1",
Attributes.StimulusFrequencyStop, out FrequencyStop)) != 0)
            goto Error;
        Console.WriteLine("Frequency Stop: " + FrequencyStop);

        if ((Status = nwa.GetAttributeViInt32("Channel1",
Attributes.StimulusPoints, out Points)) != 0)
            goto Error;
        Console.WriteLine("Points: " + Points);

        if ((Status =
nwa.MeasurementSetParameter("Channel1:Measurement1",
Constants.MeasurementTypeSParm, 1, 1)) != 0)
            goto Error;

        if ((Status = nwa.SetAttributeViInt32("Channel1:Measurement1",
Attributes.MeasurementFormat,
Constants.MeasurementFormatPolarReallmage)) != 0)
            goto Error;

        if ((Status = nwa.SetAttributeViInt32("",
Attributes.StimulusTriggerSource, Constants.TriggerSourceBus)) != 0)
```

C#

```
        goto Error;
    if((Status = nwa.TriggerSingle("")) != 0)
        goto Error;

    Frequencies = new double[Points];
    realData = new double[Points];
    imagData = new double[Points];

    if ((Status = nwa.ChannelFrequencyData("Channel1", Frequencies, out
Points)) != 0)
        goto Error;

    if ((Status =
nwa.MeasurementFetchFormatted("Channel1:Measurement1", realData, out
Points, imagData, out Points)) != 0)
        goto Error;

    Console.WriteLine("Frequencies:   Real Data:   Imag Data:");
    for (int i = 0; i < Frequencies.Length; i++)
        Console.WriteLine(" " + Frequencies[i] + "   " + realData[i] + "   " +
imagData[i]);

Error:
    if (Status != 0)
    {
        nwa.ErrorMessage(Status, out errorMessage);
        Console.Write("Error: " + errorMessage);
    }
    nwa.Close();
    Console.ReadKey();
}
}
```






```
C#
```

```
}
```



For additional programming information see the Getting Started Guide on the [Microsoft Foundation](#) website.

CMT.Instruments







This section contains a reference materials with the description of each method of the NetworkAnalyzer interface and also the description of all attributes and constants exposed from the IVI.NET driver.

	Class	Description
	Attributes	List of attributes (attribute ID-s) that control the operation of the instruments.
	Constants	List of constants whose values are assumed by members of Attributes class.
	NetworkAnalyzer	class NetworkAnalyzer

class NetworkAnalyzer










	Name	Description
	<u>NetworkAnalyzer(String, Boolean, Boolean)</u>	Opens the I/O session to the instrument.
	<u>NetworkAnalyzer(String, Boolean, Boolean, String)</u>	Opens the I/O session to the instrument with options.






Methods



	Name	Description
	<u>AbortPrint</u>	Aborts the printout.
	<u>ApplyCalibration</u>	Applies calibration to channels. This method must be executed after acquiring all the Standards during calibration session.
	<u>ApplySimplifiedCalibration</u>	<p>Calculates the calibration coefficients for the simplified 3 or 4 port calibration from the calibration standards measurements when the 3 or 4 port calibration is selected as the calibration type.</p> <p>The calibration type is selected by <u>InitUserCal</u> function.</p> <p>The simplified 3 port calibration allows to omit one THRU measurement. The simplified 4 port calibration allows to omit up to three THRU measurements. If full set of calibration standard measurement is made this command behaves like the <u>ApplyCalibration</u> function.</p>
	<u>CalKitLoadFromFile</u>	Recalls the definition file for the calibration kit. The file must be saved by the CalKitSaveToFile function.
	<u>CalKitRestore</u>	Resets the calibration kit to the factory settings. Restores the predefined calibration kit. Removes the user defined calibration kit.
	<u>CalKitSaveToFile</u>	Saves the definition file for the calibration kit.








	Name	Description
	<u>CalStandardInsert</u>	Inserts the calibration standard into the selected calibration kit. The existing standards with indices greater than or equal to inserted standard are shifted by +1.
	<u>CalStandardRemove</u>	Deletes the calibration standard into the selected calibration kit. The existing standards with indices greater than inserted standard are shifted by -1.
	<u>CancelCalibration</u>	Clears the measurement data of the calibration standards.
	<u>ChannelFrequencyData</u>	The array size is N, where N is the number of measurement points.
	<u>ClearAverage</u>	Clears and restarts the averaging process, when the averaging function is turned on.
	<u>ClearError</u>	<p>This function clears the error code and error description for the current execution thread and for the session.</p> <p>If the user specifies a valid MI session for the Vi parameter, this function clears the error information for the session.</p> <p>If the user passes VI_NULL for the Vi parameter, this function clears the error information for the current execution thread.</p> <p>If the Vi parameter is an invalid session, the function does nothing and returns an error.</p>








	Name	Description
	ClearRegisterStates	Clears the memory of the channel state saved by SaveChannelToRegister function.
	ClearRippleLimitTable	Clears the ripple limit table. The function is used for the active trace of the active channel.
	Close	<p>Closes the I/O session to the instrument.</p> <p>Driver methods and properties that access the instrument are not accessible after Close is called.</p>
	ConfigureFrequencyCenterSpan	Sets the sweep range for the channel using center and span value of the frequencies.
	ConfigureFrequencyStartStop	Sets the sweep range for the channel using start and stop value of the frequencies.
	ConfigureGatingCenterSpan	<p>Configures the center and span values of the gating function.</p> <p>This function configures the center and span points in terms of time.</p>
	ConfigureGatingStartStop	<p>Configures the start and stop values of the gating function.</p> <p>This function configures the start and stop points in terms of time.</p>
	ConfigurePowerCenterSpan	Sets the sweep range for the channel using center and span value of the output power.










	Name	Description
	ConfigurePowerStartStop	Sets the sweep range for the channel using start and stop value of the output power.
	ConfigureSweep	Configures the number of points for the sweep measurement and the sweep mode.
	ConfigureTimeDomainCenterSpan	Configures the center and span values for the time-domain analysis. This function configures the center and span points in terms of time.
	ConfigureTimeDomainStartStop	Configures the start and stop values for the time-domain analysis. This function configures the start and stop points in terms of time.
	ContinuousAllChannels	Turns ON the continuous trigger initiation mode for all channels.
	Disable	Places the instrument in a quiescent state where it has minimal or no impact on the system to which it is connected.
	DisplaySetDefaults	Restores the display settings to the default values.
	Equals	(Inherited from Object .)
	ErrorMessage	Translates the error return value from an driver function to a user-readable string. The user should pass a buffer with at least 256 bytes for the ErrorMessage parameter.







	Name	Description
	<u>ErrorQuery</u>	Queries the instrument and returns instrument specific error information. This function can be used when QueryInstrumentStatus is True to retrieve error details when the driver detects an instrument error.
	<u>ExecuteAutoCalCalibration</u>	Executes 1-port calibration of the specified port of specified channel or full 2-port calibration between 2 specified ports of specified channel or one path 2-port calibration between 2 specified ports of specified channel using the AutoCal module.
	<u>ExecuteAutoCalConfidenceCheck</u>	Executes the confidence check of the calibration coefficients of specified channel using the AutoCal module. The function sets the AutoCal Module to the special internal state, reads the S-parameters of this state from the AutoCal Module and sets memory traces so that they can be compared with actual measured data. Comparison is carried out visually by the user.
	<u>ExecuteAutoCalNPortCalibration</u>	Executes 1-port calibration of the specified port of specified channel or full 2, 3, 4-port calibration between the specified 2, 3, 4 ports of specified channel or one path 2-port calibration between the specified 2 ports of specified channel using the AutoCal module.
	<u>ExecuteAutoOrientation</u>	Executes the Auto-Orientation procedure of the AutoCal Module. The AutoCal Module must be connected to the ports of Analyzer.


	Name	Description
	<u>ExportLossTable</u>	Saves the loss compensation table into a file.
	<u>Finalize</u>	Closes the I/O session to the instrument (Overrides Object.Finalize())
	<u>FunctionExecute</u>	Executes the analysis specified for FunctionType integer Property. See more details: Properties.FunctionType
	<u>GetAttributeViBoolean</u>	Queries the value of a ViBoolean attribute
	<u>GetAttributeViInt32</u>	Queries the value of a ViInt32 attribute
	<u>GetAttributeViReal64</u>	Queries the value of a ViReal64 attribute
	<u>GetAttributeViString</u>	Queries the value of a ViString attribute
	<u>GetCalculatedFunctionData</u>	Gets the data array, which is the FunctionExecute function analysis result.
	<u>GetCalibrationInfo</u>	Gets the information string of the calibration acting between the source and receiver ports.
	<u>GetCalKitOrderLoad</u>	Gets the number of the calibration standard of the load type, used for the measurement of the specified port.
	<u>GetCalKitOrderOpen</u>	Gets the number of the calibration standard of the open type, used for









	Name	Description
		the measurement of the specified port.
	<u>GetCalKitOrderShort</u>	Gets the number of the calibration standard of the short type, used for the measurement of the specified port.
	<u>GetCalKitOrderThru</u>	Gets the number of the calibration standard of the thru type, used for the measurement between the source and receiver ports.
	<u>GetCalKitOrderTrlLine</u>	Gets the number of the calibration standard of the TRL line type, used for the measurement between the source and receiver ports.
	<u>GetCalKitOrderTrlReflect</u>	Gets the number of the calibration standard of the TRL Reflect type, used for the measurement of the specified port.
	<u>GetCalKitOrderTrlThru</u>	Gets the number of the calibration standard of the TRL thru type, used for the measurement between the source and receiver ports.
	<u>GetCalStandardS1PData</u>	Gets the data array of the data-based calibration standard. The first element of the array is 1 and determines the number of ports of the calibration standard.
	<u>GetCalStandardS2PData</u>	Gets the data array of the data-based calibration standard. The first element of the array is 2 and determines the number of ports of the calibration standard.







	Name	Description
	GetError	<p>This function retrieves and then clears the IVI error information for the session or the current execution thread.</p> <p>If the user specifies a valid IVI session for the Vi parameter, GetError retrieves and then clears the error information for the session.</p> <p>If the user passes VI_NULL for the Vi parameter, GetError retrieves and then clears the error information for the current execution thread.</p> <p>If the Vi parameter is an invalid session, the function does nothing and returns an error.</p> <p>Normally, the error information describes the first error that occurred since the user last called GetError or ClearError function.</p>
	GetHashCode	(Inherited from Object)
	GetLimitTestData	Gets the data array, which is the limit line in the limit test function.
	GetLimitTestReport	Gets the data array, which is the stimulus values of the measurement points that failed the limit test.
	GetLimitTestReportAll	Gets the data array, which is the limit test result.
	GetLimitTestStatus	Gets the limit test result.
	GetMarkerValue	Gets the response value of the marker. If the reference marker is turned ON, the values of the markers




	Name	Description
		from 1 to 15 are read out as relative values to the reference marker.
	GetMaxChannelCount	Gets the maximum number of channels.
	GetMaxTraceCount	Gets the maximum number of traces in the channel.
	GetPowerCalibrationTable	<p>Gets the power correction array (result of power calibration executed by <code>PerformCalibrationAction</code> function).</p> <p>The array size is NOP, where NOP is the number of measurement points.</p>
	GetPowerLossCompensationTable	Gets the loss compensation table used when the power calibration is executed by <code>PerformCalibrationAction</code> function.
	GetRippleLimitData	Gets the data array, which is the limit line for the ripple limit function.
	GetRippleLimitTestReport	Gets the data array, which is the ripple limit test result.
	GetRippleLimitTestStatus	Gets the ripple limit test result.
	GetSegmentData	Gets the array of the segment sweep table.
	GetSubclassStdOrder	<p>Gets the subclass used to specify classes of calibration standards by functions:</p> <p>NetworkAnalyzer.GetCalKitOrderOpen</p>

	Name	Description
		NetworkAnalyzer.GetCalKitOrderShort NetworkAnalyzer.GetCalKitOrderLoad NetworkAnalyzer.GetCalKitOrderThru NetworkAnalyzer.GetCalKitOrderTrlLine NetworkAnalyzer.GetCalKitOrderTrlThru NetworkAnalyzer.GetCalKitOrderTrlReflect
	GetTouchstoneFileType	Gets the Touchstone file type and the port numbers, when saving S-parameters by SaveTouchstoneFile function.
	GetType	(Inherited from Object .)
	HoldAllChannels	Turns OFF the continuous trigger initiation mode for all channels.
	ImportLossTable	Recalls the loss compensation file. The file must be saved by the ExportLossTable function.
	InitNPortUserCal	Initializes N-port user calibration.
	InitUserCal	Initializes user calibration.
	LoadTouchstoneFile	Loads the Touchstone file with the specified name to the measured S-parameters of the active channel.

	Name	Description
		<p>The Touchstone file types 1, 2, 3 or 4 port (file extensions s1p, s2p, s3p or s4p) are supported.</p> <p>On completion of the command, the channel goes to the hold state.</p>
	LockSession	Obtains a multithread lock on the instrument session.
	MarkerFunctionExecute	<p>Executes the marker search according to the specified criterion.</p> <p>The type of the marker search is set by MarkerSearchType</p>
	MarkerFunctionsMarkerCenter	<p>Sets the value of the specified item to the value of the position of the marker.</p> <p>Sweep center value is set to the stimulus value of the marker position.</p>
	MarkerFunctionsMarkerDelay	<p>Sets the value of the specified item to the value of the position of the marker.</p> <p>Delay value is set to the response value of the marker position.</p>
	MarkerFunctionsMarkerRefValue	<p>Sets the value of the specified item to the value of the position of the marker.</p> <p>Reference value is set to the response value of the marker position.</p>


	Name	Description
	<u>MarkerFunctionsMarkerStart</u>	<p>Sets the value of the specified item to the value of the position of the marker.</p> <p>Sweep start value is set to the stimulus value of the marker position.</p>
	<u>MarkerFunctionsMarkerStop</u>	<p>Sets the value of the specified item to the value of the position of the marker.</p> <p>Sweep stop value is set to the stimulus value of the marker position.</p>
	<u>MarkerSetLimitLineResponseOffset</u>	Sets the value of the limit line offset along Y-axis to the active marker value.
	<u>MeasurementAutoRefValue</u>	Executes the auto reference function for the trace. The function automatically sets the reference level value.
	<u>MeasurementAutoScale</u>	Executes the auto scale function for the trace.
	<u>MeasurementDataToMemory</u>	Saves measurement trace data in memory.
	<u>MeasurementFetchComplex</u>	Returns the corrected data array. The corrected data array is the data, which processing is completed excluding the formatting as the last step. Such data represent S-parameter complex values.
	<u>MeasurementFetchFormatted</u>	Returns the formatted data array. The formatted data array is the data, which processing is

	Name	Description
		completed including the formatting as the last step. Such data represent the data trace values as they are shown on the screen.
	MeasurementFetchMemoryComplex	Returns the corrected memory array. The corrected memory array is the data, which processing is completed excluding the formatting as the last step. Such data represent S-parameter complex values.
	MeasurementFetchMemoryFormatted	Returns the formatted memory array. The formatted memory array is the data, which processing is completed including the formatting as the last step. Such data represent the memory trace values as they are shown on the screen.
	MeasurementFetchX	Gets the X-axis values array. The X-axis values array is the frequency, power or time values array depending on the trace setup. The array contains real values. The array size is N, where N is the number of measurement points.
	MeasurementGetParameter	Gets the measurement parameter of the trace.
	MeasurementSetParameter	Sets the measurement parameter of the trace.
	MeasurePortExtensionOpen	<p>Performs measurement of the standard "OPEN", automatically calculates and sets the parameters of the Port Extension.</p> <p>When two consecutive measurements of "OPEN" is</p>









	Name	Description
		performed the results of these measurements are averaged.
	MeasurePortExtensionShort	<p>Performs measurement of the standard "SHORT" automatically calculates and sets the parameters of the Port Extension.</p> <p>When two consecutive measurements of "SHORT" is performed the results of these measurements are averaged.</p>
	MemberwiseClone	(Inherited from Object .)
	PerformCalibrationAction	Performs one calibration action on the specified ports or the type of calibration action. It is recommended to use WaitForOperationComplete function to check if operation is completed.
	PerformUnkThru2PortCal	<p>Completes the full 2-port calibration between the specified ports provided so that each port was calibrated using full 1-port calibration:</p> <ul style="list-style-type: none"> • Measures an arbitrary thru between the ports; • Calculates the error terms Et and Ei using the unknown thru algorithm; • Saves the Et and Ei error terms to the existing calibration getting the full 2-port calibration from the two 1-port calibrations. If the full 2-port calibration already existed between the specified ports, updates the Et and Ei error terms. It is recommended to use

	Name	Description
		CmtNA_WaitForOperationComplete function to check if the operation is completed.
	PowerSensorZeroing	<p>Executes zeroing procedure of the power sensor.</p> <p>Although the Analyzer automatically turns off the RF power during this procedure, it is recommended to disconnect the power sensor from the analyzer port.</p>
	PrintOut	Prints out the image displayed on the screen without previewing.
	QueryMarkerMathBandwidth	<p>Gets the bandwidth search result.</p> <p>The bandwidth search can be performed relatively to the marker, or relatively to the absolute maximum value of the trace (in this case the number of the marker is ignored), that is set by MarkerMathBandwidthSearchRef</p>
	QueryMarkerMathFlatness	Gets FLATNESS function data array. The FLATNESS function is applied within the range determined by two markers.
	QueryMarkerMathStatistics	<p>Gets the math statistics values.</p> <p>The statistics function is applied either over the whole range, or within the range specified by MarkerMathStatistics (the range limits are determined by two markers).</p>
	QueryMarkerTableData	Gets the data array of all turned ON markers.

	Name	Description
		<p>The array size is $3N + 1$, where N is the number of turned ON markers including the reference marker.</p> <p>If the reference marker is turned ON the last three elements of the array contain the reference marker data and the rest elements of array contain the relative values.</p>
	RecallChannelFromRegister	<p>Recalls the Analyzer state for the active channel.</p> <p>The file must be saved in one of the four memory registers by SaveChannelToRegister function.</p>
	RecallFromFile	<p>Recalls the specified Analyzer state file.</p> <p>The file must be saved by SaveStateToFile function.</p>
	RecallSegmentTable	Recalls the segment table file.
	Reset	Resets the device to a known initialization state.
	ResetCalibration	Clears the calibration coefficient table.
	ResetWithDefaults	Resets the device to a known initialization state with defaults as specified for the device in the optionString parameter.
	RestoreLimitTable	Recalls the limit table file. The file must be saved by SaveLimitTable function.





	Name	Description
		The function is used for the active trace of the active channel.
	RestoreRippleLimitTable	<p>Recalls the ripple limit table file.</p> <p>The file must be saved by SaveRippleLimitTable function.</p> <p>The function is used for the active trace of the active channel.</p>
	RevisionQuery	Queries the revision of the device.
	SaveChannelToRegister	Saves the Analyzer state of the items set for the active channel into one of the four memory registers.
	SaveImage	Saves the display image in BMP or PNG format into a file.
	SaveLimitTable	<p>Recalls the limit table file.</p> <p>The function is used for the active trace of the active channel.</p>
	SaveRippleLimitTable	<p>Saves the ripple limit table into a file.</p> <p>The function is used for the active trace of the active channel.</p>
	SaveSegmentTable	Saves the segment table in a file.
	SaveStateToFile	Saves the Analyzer state into a file.
	SaveTouchstoneFile	Saves the measured S-parameters of the active channel into a Touchstone file.






	Name	Description
		The file type (1 port to 4 port) is set by SetTouchstoneFileType function.
	SaveTraceData	<p>Saves the CSV formatted data into a file.</p> <p>The function is used for the active trace of the active channel.</p>
	SelfTest	Performs an instrument self test, waits for the instrument to complete the test, and queries the instrument for the results.
	SetAttributeViBoolean	Sets the value of a ViBoolean attribute
	SetAttributeViInt32	Sets the value of a ViInt32 attribute.
	SetAttributeViReal64	Sets the value of a ViReal64 attribute
	SetAttributeViString	Sets the value of a ViString attribute.
	SetCalKitOrderLoad	Sets the number of the calibration standard of the load type, used for the measurement of the specified port.
	SetCalKitOrderOpen	Sets the number of the calibration standard of the open type, used for the measurement of the specified port.
	SetCalKitOrderShort	Sets the number of the calibration standard of the short type, used for the measurement of the specified port.

	Name	Description
	<u>SetCalKitOrderThru</u>	Sets the number of the calibration standard of the thru type, used for the measurement between the source and receiver ports.
	<u>SetCalKitOrderTrlLine</u>	Sets the number of the calibration standard of the TRL line type, used for the measurement between the source and receiver ports.
	<u>SetCalKitOrderTrlReflect</u>	Sets the number of the calibration standard of the TRL Reflect type, used for the measurement of the specified port.
	<u>SetCalKitOrderTrlThru</u>	Sets the number of the calibration standard of the TRL thru type, used for the measurement between the source and receiver ports.
	<u>SetCalStandardS1PData</u>	Sets the data array of the data-based calibration standard. The first element of the array is 1 and determines the number of ports of the calibration standard.
	<u>SetCalStandardS2PData</u>	Sets the data array of the data-based calibration standard. The first element of the array is 2 and determines the number of ports of the calibration standard.
	<u>SetLimitTestData</u>	Sets the data array, which is the limit line in the limit test function.
	<u>SetPowerCalibrationTable</u>	Sets the power correction array (result of power calibration executed by PerformCalibrationAction function).

	Name	Description
		The array size is NOP, where NOP is the number of measurement points.
	SetPowerLossCompensationTable	Sets the loss compensation table used when the power calibration is executed by PerformCalibrationAction function.
	SetRippleLimitData	Sets the data array, which is the limit line for the ripple limit function.
	SetSegmentData	Sets the array of the segment sweep table.
	SetSubclassStdOrder	<p>Sets the subclass used to specify classes of calibration standards by functions:</p> <p>NetworkAnalyzer.SetCalKitOrderOpen</p> <p>NetworkAnalyzer.SetCalKitOrderShort</p> <p>NetworkAnalyzer.SetCalKitOrderLoad</p> <p>NetworkAnalyzer.SetCalKitOrderThru</p> <p>NetworkAnalyzer.SetCalKitOrderTrlLine</p> <p>NetworkAnalyzer.SetCalKitOrderTrlThru</p> <p>NetworkAnalyzer.SetCalKitOrderTrlReflect</p>

	Name	Description
	<u>SetTouchstoneFileType</u>	Sets the Touchstone file type and the port numbers, when saving S-parameters by <u>SaveTouchstoneFile</u> function.
	<u>SystemHide</u>	Minimizes the analyzer main window removing it from the desktop.
	<u>SystemPreset</u>	Resets the Analyzer to the factory settings.
	<u>SystemShow</u>	Restores the analyzer main window hidden by <u>SystemHide</u> function.
	<u>TakePowerCalSweep</u>	<p>Measures the power calibration data for the port using the power meter controlled via USB or USB/GPIB.</p> <p>Calculates calibration coefficients on completion of the measurement, and turns ON the power correction for the port.</p>
	<u>TestBeepComplete</u>	Generates a beep to notify of the completion of the operation.
	<u>TestBeepWarning</u>	Generates a beep to notify of warning.
	<u>TimeDomainSetFrequencyLowPass</u>	Changes the frequency range to match with the low-pass type of the time domain transformation function.
	ToString	(Inherited from Object .)
	<u>TraceHoldRestart</u>	This command resets the trace hold function.

	Name	Description
	<u>TriggerImmediate</u>	Generates a trigger signal and initiates a sweep under the following conditions: - Trigger source is set to the BUS (set by the command TRIG:SOUR BUS), otherwise an error occurs and the command is ignored. - Analyzer must be in the trigger waiting state, otherwise (the analyzer is in the measurement state or in the hold state) an error occurs and the function is ignored. The function is completed immediately after the generation of the trigger signal (does not wait the end of a sweep).
	<u>TriggerInit</u>	<p>Puts the channel to the Trigger Waiting state for the one trigger event.</p> <p>The channel should be in the hold state, otherwise an error occurs and the command is ignored.</p>
	<u>TriggerRestart</u>	<p>Aborts the sweep. The channels in the Single trigger initiation mode transit to the Hold state.</p> <p>The channels in the Continuous trigger initiation mode transit to the trigger waiting state, if the trigger source is set to Internal, the channel immediately starts a new sweep.</p>
	<u>TriggerSingle</u>	Generates a trigger signal and initiates a sweep under the following conditions: - Trigger source is set to the BUS (set by the CMTNA_ATTR_STIMULUS_TRIGGER_SOURCE), otherwise an error occurs and the function is ignored. - Analyzer must be in the trigger waiting state, otherwise (the

	Name	Description
		analyzer is in the measurement state or in the hold state) an error occurs and the function is ignored.
	TriggerSingleAndWait	Generates a trigger signal and initiates a sweep under the following conditions: - Trigger source is set to the BUS (set by the CMTNA_ATTR_STIMULUS_TRIGGER_SOURCE), otherwise an error occurs and the function is ignored. - Analyzer must be in the trigger waiting state, otherwise (the analyzer is in the measurement state or in the hold state) an error occurs and the function is ignored.
	UnlockSession	Releases a lock obtained on an device driver session by calling the CmtNA_LockSession function.
	WaitForOperationComplete	Method returns when all pending operations are complete or maxTimeMilliseconds exceeded.
	WaitForSweepComplete	Wait the end of the sweep.
	WaitForTriggerState	Waits the specified state of the analyzer has been reached.

AbortPrint

Description

Aborts the printout.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Syntax

Visual Basic (Declaration)

```
Public Function AbortPrint As Integer
```

C#

```
public int AbortPrint()
```

Visual C++

```
public:  
int AbortPrint()
```

JavaScript

```
function abortPrint();
```

Return Value

Success or failure code.

Back to [Methods](#)

ApplyCalibration

Description

Applies calibration to channels. This method must be executed after acquiring all the Standards during calibration session.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Syntax

Visual Basic (Declaration)

```
Public Function ApplyCalibration ( _  
    repeatedCapabilitiesID As String _  
) As Integer
```

C#

```
public int ApplyCalibration(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int ApplyCalibration(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function applyCalibration(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Return Value

Success or failure code.

Back to [Methods](#)

ApplySimplifiedCalibration

Description

Calculates the calibration coefficients for the simplified 3 or 4 port calibration from the calibration standards measurements when the 3 or 4 port calibration is selected as the calibration type.

The calibration type is selected by [InitUserCal](#) function.

The simplified 3 port calibration allows to omit one THRU measurement. The simplified 4 port calibration allows to omit up to three THRU measurements. If full set of calibration standard measurement is made this command behaves like the [ApplyCalibration](#) function.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function ApplySimplifiedCalibration ( _  
    repeatedCapabilitiesID As String _  
    ) As Integer
```

C#

```
public int ApplySimplifiedCalibration(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int ApplySimplifiedCalibration(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function applySimplifiedCalibration(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Return Value

Success or failure code.

Back to [Methods](#)

CalKitLoadFromFile

Description

Recalls the definition file for the calibration kit. The file must be saved by the CalKitSaveToFile function.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function CalKitLoadFromFile ( _  
    kitIndex As Integer, _  
    fileName As String _  
) As Integer
```

C#

```
public int CalKitLoadFromFile(  
    int kitIndex,  
    string fileName  
)
```

Visual C++

```
public:  
int CalKitLoadFromFile(  
    int kitIndex,  
    String fileName  
)
```

JavaScript

```
function calKitLoadFromFile(kitIndex, fileName);
```

Parameters

kitIndex

Type: **System.Int32**

The index of the calibration kit.

fileName

Type: **System.String**

The file path into which to save the data. If the full path of the file is not specified, the \CalKit subdirectory of the application directory will be searched for the file. The calibration kit definition file has *.ckd extension by default.

Return Value

Success or failure code.

Back to [Methods](#)

CalKitRestore

Description

Resets the calibration kit to the factory settings. Restores the predefined calibration kit. Removes the user defined calibration kit.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function CalKitRestore As Integer
```

C#

```
public int CalKitRestore()
```

Visual C++

```
public:  
int CalKitRestore()
```

JavaScript

```
function calKitRestore();
```

Return Value

Success or failure code.

Back to [Methods](#)

CalKitSaveToFile

Description

Saves the definition file for the calibration kit.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function CalKitSaveToFile ( _  
    kitIndex As Integer, _  
    fileName As String _  
) As Integer
```

C#

```
public CalKitSaveToFile(  
    int kitIndex,  
    string fileName  
)
```

Visual C++

```
public:  
int CalKitSaveToFile(  
    int kitIndex,  
    String fileName  
)
```

JavaScript

```
function calKitSaveToFile(kitIndex, fileName);
```

Parameters

kitIndex

Type: **System.Int32**

The index of the calibration kit.

fileName

Type: **System.String**

The file path into which to save the data. If the full path of the file is not specified, the \CalKit subdirectory of the application directory will be searched for the file. The calibration kit definition file has *.ckd extension by default.

Return Value

Success or failure code.

Back to [Methods](#)

CalStandardInsert

Description

Inserts the calibration standard into the selected calibration kit. The existing standards with indices greater than or equal to inserted standard are shifted by +1.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function CalStandardInsert ( _  
    repeatedCapabilitiesID As String _  
) As Integer
```

C#

```
public CalStandardInsert(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int CalStandardInsert(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function calStandardInsert(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.

Return Value

Success or failure code.

Back to [Methods](#)

CalStandardRemove

Description

Deletes the calibration standard into the selected calibration kit. The existing standards with indices greater than inserted standard are shifted by -1 .

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function CalStandardRemove ( _  
    repeatedCapabilitiesID As String _  
) As Integer
```

C#

```
public CalStandardRemove(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int CalStandardRemove(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function CalStandardRemove(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.

Return Value

Success or failure code.

Back to [Methods](#)

CancelCalibration

Description

Clears the measurement data of the calibration standards.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function CancelCalibration ( _  
    repeatedCapabilitiesID As String _  
) As Integer
```

C#

```
public int CancelCalibration(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int CancelCalibration(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function cancelCalibration(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Return Value

Success or failure code.

Back to [Methods](#)

ChannelFrequencyData

Description

The array size is N, where N is the number of measurement points.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function ChannelFrequencyData ( _  
    repeatedCapabilitiesID As String, _  
    retValues As Double(), _  
    <OutAttribute> ByRef returnSize As Integer _  
) As Integer
```

C#

```
public int ChannelFrequencyData(  
    string repeatedCapabilitiesID,  
    double[] retValues,  
    out int returnSize  
)
```

Visual C++

```
public:  
int ChannelFrequencyData(  
    String repeatedCapabilitiesID,  
    double retValues[],  
    int *returnSize  
)
```

JavaScript

```
function channelFrequencyData(repeatedCapabilitiesID, retValues,  
returnSize);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

retValues

Type: **System.Double** []

The array of the frequency value at the n–th measurement point.

returnSize

Type: **System.Int32**

The actual number of elements filled by the query.

Return Value

Success or failure code.

Back to [Methods](#)

ClearAverage

Description

Clears and restarts the averaging process, when the averaging function is turned on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function ClearAverage (_  
    repeatedCapabilitiesID As String _  
) As Integer
```

C#

```
public int ClearAverage(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int ClearAverage(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function clearAverage(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Return Value

Success or failure code.

Back to [Methods](#)

ClearError

Description

This function clears the error code and error description for the current execution thread and for the session.

If the user specifies a valid VI session for the Vi parameter, this function clears the error information for the session.

If the user passes VI_NULL for the Vi parameter, this function clears the error information for the current execution thread.

If the Vi parameter is an invalid session, the function does nothing and returns an error.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function ClearError As Integer
```

C#

```
public int ClearError()
```

Visual C++

```
public:  
int ClearError()
```

JavaScript

```
function clearError();
```

Return Value

Success or failure code.

Back to [Methods](#)

ClearRegisterStates

Description

Clears the memory of the channel state saved by [SaveChannelToRegister](#) function.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function ClearRegisterStates ( _  
    repeatedCapabilitiesID As String _  
) As Integer
```

C#

```
public int ClearRegisterStates(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int ClearRegisterStates(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function clearRegisterStates(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Return Value

Success or failure code.

Back to [Methods](#)

ClearRippleLimitTable

Description

Clears the ripple limit table. The function is used for the active trace of the active channel.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function ClearRippleLimitTable ( _  
    repeatedCapabilitiesID As String _  
) As Integer
```

C#

```
public int ClearRippleLimitTable(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int ClearRippleLimitTable(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function clearRippleLimitTable(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Return Value

Success or failure code.

Back to [Methods](#)

Close

Description

Closes the I/O session to the instrument.

Driver methods and properties that access the instrument are not accessible after Close is called.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Sub Close
```

C#

```
public void Close()
```

Visual C++

```
public:  
void Close()
```

JavaScript

```
function close();
```

Return Value

Success or failure code.

Back to [Methods](#)

ConfigureFrequencyCenterSpan

Description

Sets the sweep range for the channel using center and span value of the frequencies.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function ConfigureFrequencyCenterSpan ( _  
    repeatedCapabilitiesID As String, _  
    centerFrequency As Double, _  
    frequencySpan As Double _  
) As Integer
```

C#

```
public int ConfigureFrequencyCenterSpan(  
    string repeatedCapabilitiesID,  
    double centerFrequency,  
    double frequencySpan  
)
```

Visual C++

```
public:  
int ConfigureFrequencyCenterSpan(  
    String repeatedCapabilitiesID,  
    double centerFrequency,  
    double frequencySpan  
)
```

JavaScript

```
function configureFrequencyCenterSpan(repeatedCapabilitiesID,  
centerFrequency, frequencySpan);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

centerFrequency

Type: **System.Double**

The center frequency in a frequency sweep.

frequencySpan

Type: **System.Double**

The frequency span for the frequency sweep.

Return Value

Success or failure code.

Back to [Methods](#)

ConfigureFrequencyStartStop

Description

Sets the sweep range for the channel using start and stop value of the frequencies.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function ConfigureFrequencyStartStop ( _  
    repeatedCapabilitiesID As String, _  
    startFrequency As Double, _  
    stopFrequency As Double _  
) As Integer
```

C#

```
public int ConfigureFrequencyStartStop(  
    string repeatedCapabilitiesID,  
    double startFrequency,  
    double stopFrequency  
)
```

Visual C++

```
public:  
int ConfigureFrequencyStartStop(  
    String repeatedCapabilitiesID,  
    double startFrequency,  
    double stopFrequency  
)
```

JavaScript

```
function configureFrequencyStartStop(repeatedCapabilitiesID,  
startFrequency, stopFrequency);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

startFrequency

Type: **System.Double**

The lower limit of a span of frequencies.

stopFrequency

Type: **System.Double**

The upper limit of a span of frequencies.

Return Value

Success or failure code.

Back to [Methods](#)

ConfigureGatingCenterSpan

Description

Configures the center and span values of the gating function.

This function configures the center and span points in terms of time.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function ConfigureGatingCenterSpan ( _  
    repeatedCapabilitiesID As String, _  
    gatingCenter As Double, _  
    gatingSpan As Double, _  
    type As Integer _  
) As Integer
```

C#

```
public int ConfigureGatingCenterSpan(  
    string repeatedCapabilitiesID,  
    double gatingCenter,  
    double gatingSpan,  
    int type  
)
```

Visual C++

```
public:  
int ConfigureGatingCenterSpan(  
    String repeatedCapabilitiesID,  
    double gatingCenter,
```

Visual C++

```
double gatingSpan,  
int type  
)
```

JavaScript

```
function configureGatingCenterSpan(repeatedCapabilitiesID, gatingCenter,  
gatingSpan, type);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

gatingCenter

Type: **System.Double**

The gating function center value.

gatingSpan

Type: **System.Double**

The gating function span value.

type

Type: **System.Int32**

The gating function type.

[TimeDomainGatingBandpass](#)

[TimeDomainGatingNotch](#)

Return Value

Success or failure code.

Back to [Methods](#)

ConfigureGatingStartStop

Description

Configures the start and stop values of the gating function.

This function configures the start and stop points in terms of time.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function ConfigureGatingStartStop ( _  
    repeatedCapabilitiesID As String, _  
    gatingStart As Double, _  
    gatingStop As Double, _  
    type As Integer _  
) As Integer
```

C#

```
public int ConfigureGatingStartStop(  
    string repeatedCapabilitiesID,  
    double gatingStart,  
    double gatingStop,  
    int type  
)
```

Visual C++

```
public:  
int ConfigureGatingStartStop(  
    String repeatedCapabilitiesID,  
    double gatingStart,
```

Visual C++

```
double gatingStop,  
int type  
)
```

JavaScript

```
function configureGatingStartStop(repeatedCapabilitiesID, gatingstart,  
gatingStop, type);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

gatingStart

Type: **System.Double**

The gating function start value.

gatingStop

Type: **System.Double**

The gating function stop value.

type

Type: **System.Int32**

The gating function type.

[TimeDomainGatingBandpass](#)

[TimeDomainGatingNotch](#)

Return Value

Success or failure code.

Back to [Methods](#)

ConfigurePowerCenterSpan

Description

Sets the sweep range for the channel using center and span value of the output power.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function ConfigurePowerCenterSpan ( _  
    repeatedCapabilitiesID As String, _  
    centerPower As Double, _  
    spanPower As Double _  
    ) As Integer
```

C#

```
public int ConfigurePowerCenterSpan(  
    string repeatedCapabilitiesID,  
    double centerPower,  
    double spanPower  
)
```

Visual C++

```
public:  
int ConfigurePowerCenterSpan(  
    String repeatedCapabilitiesID,  
    double centerPower,  
    double spanPower  
)
```

JavaScript

```
function configurePowerCenterSpan(repeatedCapabilitiesID, centerPower,  
spanPower);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

centerPower

Type: **System.Double**

The center value of the output power for power sweep.

spanPower

Type: **System.Double**

The span value of the output power for power sweep.

Return Value

Success or failure code.

Back to [Methods](#)

ConfigurePowerStartStop

Description

Sets the sweep range for the channel using start and stop value of the output power.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function ConfigurePowerStartStop ( _  
    repeatedCapabilitiesID As String, _  
    startPower As Double, _  
    stopPower As Double _  
) As Integer
```

C#

```
public int ConfigurePowerStartStop(  
    string repeatedCapabilitiesID,  
    double startPower,  
    double stopPower  
)
```

Visual C++

```
public:  
int ConfigurePowerStartStop(  
    String repeatedCapabilitiesID,  
    double startPower,  
    double stopPower  
)
```

JavaScript

```
function configurePowerStartStop(repeatedCapabilitiesID, startPower,  
stopPower);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

startPower

Type: **System.Double**

The start power for power sweep.

stopPower

Type: **System.Double**

The stop power for power sweep.

Return Value

Success or failure code.

Back to [Methods](#)

ConfigureSweep

Description

Configures the number of points for the sweep measurement and the sweep mode.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function ConfigureSweep ( _  
    repeatedCapabilitiesID As String, _  
    numberOfPoints As Integer, _  
    sweepMode As Integer _  
) As Integer
```

C#

```
public int ConfigureSweep(  
    string repeatedCapabilitiesID,  
    int numberOfPoints,  
    int sweepMode  
)
```

Visual C++

```
public:  
int ConfigureSweep(  
    String repeatedCapabilitiesID,  
    int numberOfPoints,  
    int sweepMode  
)
```

JavaScript

```
function configureSweep(repeatedCapabilitiesID, numberOfPoints,  
sweepMode);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

numberOfPoints

Type: **System.Int32**

The number of points to acquire as part of the sweep measurement.

sweepMode

Type: **System.Int32**

The mode for the sweep measurement.

Return Value

Success or failure code.

Back to [Methods](#)

ConfigureTimeDomainCenterSpan

Description

Configures the center and span values for the time-domain analysis.

This function configures the center and span points in terms of time.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function ConfigureTimeDomainCenterSpan ( _  
    repeatedCapabilitiesID As String, _  
    centerPoint As Double, _  
    span As Double _  
) As Integer
```

C#

```
public int ConfigureTimeDomainCenterSpan(  
    string repeatedCapabilitiesID,  
    double centerPoint,  
    double span  
)
```

Visual C++

```
public:  
int ConfigureTimeDomainCenterSpan(  
    String repeatedCapabilitiesID,  
    double centerPoint,  
    double span  
)
```

JavaScript

```
function configureTimeDomainCenterSpan(repeatedCapabilitiesID,  
    centerPoint, span);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

centerPoint

Type: **System.Double**

The time domain center value.

span

Type: **System.Double**

The time domain span value.

Return Value

Success or failure code.

Back to [Methods](#)

ConfigureTimeDomainStartStop

Description

Configures the start and stop values for the time-domain analysis. This function configures the start and stop points in terms of time.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function ConfigureTimeDomainStartStop ( _  
    repeatedCapabilitiesID As String, _  
    startPoint As Double, _  
    stopPoint As Double _  
    ) As Integer
```

C#

```
public int ConfigureTimeDomainStartStop(  
    string repeatedCapabilitiesID,  
    double startPoint,  
    double stopPoint  
)
```

Visual C++

```
public:  
int ConfigureTimeDomainStartStop(  
    String repeatedCapabilitiesID,  
    double startPoint,  
    double stopPoint  
)
```

JavaScript

```
function configureTimeDomainStartStop(repeatedCapabilitiesID, startPoint,  
stopPoint);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

startPoint

Type: **System.Double**

The time domain start value.

stopPoint

Type: **System.Double**

The time domain stop value.

Return Value

Success or failure code.

Back to [Methods](#)

ContinuousAllChannels

Description

Turns ON the continuous trigger initiation mode for all channels.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function ContinuousAllChannels ( _  
    repeatedCapabilitiesID As String _  
) As Integer
```

C#

```
public int ContinuousAllChannels(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int ContinuousAllChannels(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function continuousAllChannels(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Return Value

Success or failure code.

Back to [Methods](#)

Disable

Description

Places the instrument in a quiescent state where it has minimal or no impact on the system to which it is connected.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function Disable As Integer
```

C#

```
public int Disable()
```

Visual C++

```
public:  
int Disable()
```

JavaScript

```
function disable();
```

Return Value

Success or failure code.

Back to [Methods](#)

DisplaySetDefaults

Description

Restores the display settings to the default values.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function DisplaySetDefaults As Integer
```

C#

```
public int DisplaySetDefaults()
```

Visual C++

```
public:  
    int DisplaySetDefaults()
```

JavaScript

```
function displaySetDefaults();
```

Return Value

Success or failure code.

Back to [Methods](#)

ErrorMessage

Description

Translates the error return value from an driver function to a user-readable string.

The user should pass a buffer with at least 256 bytes for the ErrorMessage parameter.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function ErrorMessage ( _  
    errorCode As Integer, _  
    <OutAttribute> ByRef errorMessage As String _  
    ) As Integer
```

C#

```
public int ErrorMessage(  
    int errorCode,  
    out string errorMessage  
)
```

Visual C++

```
public:  
int ErrorMessage(  
    int errorCode,  
    String errorMessage[]  
)
```

JavaScript

```
function errorMessage(errorCode, errorMessage);
```

Parameters

errorCode

Type: **System.Int32**

Passes the `returnValue` parameter that is returned from any driver function. The default value is 0 (VI_SUCCESS).

errorMessage

Type: **System.String**

Returns the user-readable message string that corresponds to the status code you specify.

Return Value

Success or failure code.

Back to [Methods](#)

ErrorQuery

Description

Queries the instrument and returns instrument specific error information. This function can be used when QueryInstrumentStatus is True to retrieve error details when the driver detects an instrument error.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function ErrorQuery ( _  
    <OutAttribute> ByRef errorCode As Integer, _  
    <OutAttribute> ByRef errorMessage As String _  
    ) As Integer
```

C#

```
public int ErrorQuery(  
    out int errorCode,  
    out string errorMessage  
)
```

Visual C++

```
public:  
    int ErrorQuery(  
        int *errorCode,  
        String errorMessage[]  
    )
```

JavaScript

```
function errorQuery(errorCode, errorMessage);
```

Parameters

errorCode

Type: **System.Int32**

Passes the `returnValue` parameter that is returned from any driver function. The default value is 0 (VI_SUCCESS).

errorMessage

Type: **System.String**

Returns the user-readable message string that corresponds to the status code you specify.

Return Value

Success or failure code.

Back to [Methods](#)

ExecuteAutoCalCalibration

Description

Executes 1-port calibration of the specified port of specified channel or full 2-port calibration between 2 specified ports of specified channel or one path 2-port calibration between 2 specified ports of specified channel using the AutoCal module.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function ExecuteAutoCalCalibration ( _  
    repeatedCapabilitiesID As String, _  
    autoCalType As Integer, _  
    receiverPort As Integer, _  
    sourcePort As Integer _  
) As Integer
```

C#

```
public int ExecuteAutoCalCalibration(  
    string repeatedCapabilitiesID,  
    int autoCalType,  
    int receiverPort,  
    int sourcePort  
)
```

Visual C++

```
public:  
    int ExecuteAutoCalCalibration(  
        String repeatedCapabilitiesID,  
        int autoCalType,
```

Visual C++

```
int receiverPort,  
int sourcePort  
)
```

JavaScript

```
function executeAutoCalCalibration(repeatedCapabilitiesID, autoCalType,  
receiverPort, sourcePort);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

autoCalType

Type: **System.Int32**

The type AutoCal Module calibration:

[Autocal1port](#)

[Autocal2port](#)

[AutocalOnePath2port](#)

receiverPort

Type: **System.Int32**

The number of port from which the measurement is performed.

sourcePort

Type: **System.Int32**

The number of port which is the source.

Return Value

Success or failure code.

Back to [Methods](#)

ExecuteAutoCalConfidenceCheck

Description

Executes the confidence check of the calibration coefficients of specified channel using the AutoCal module. The function sets the AutoCal Module to the special internal state, reads the S-parameters of this state from the AutoCal Module and sets memory traces so that they can be compared with actual measured data. Comparison is carried out visually by the user.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function ExecuteAutoCalConfidenceCheck ( _  
    repeatedCapabilitiesID As String _  
) As Integer
```

C#

```
public int ExecuteAutoCalConfidenceCheck(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
    int ExecuteAutoCalConfidenceCheck(  
        String repeatedCapabilitiesID  
    )
```

JavaScript

```
function executeAutoCalConfidenceCheck(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Return Value

Success or failure code.

Back to [Methods](#)

ExecuteAutoCalNPortCalibration

Description

Executes 1-port calibration of the specified port of specified channel or full 2, 3, 4-port calibration between the specified 2, 3, 4 ports of specified channel or one path 2-port calibration between the specified 2 ports of specified channel using the AutoCal module.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function ExecuteAutoCalNPortCalibration ( _  
    repeatedCapabilitiesID As String, _  
    autoCalType As Integer, _  
    portList As Integer() _  
) As Integer
```

C#

```
public int ExecuteAutoCalNPortCalibration(  
    string repeatedCapabilitiesID,  
    int autoCalType,  
    int[] portList  
)
```

Visual C++

```
public:  
int ExecuteAutoCalNPortCalibration(  
    String repeatedCapabilitiesID,  
    int autoCalType,  
    int portList[]  
)
```

JavaScript

```
function executeAutoCalNPortCalibration(repeatedCapabilitiesID,  
autoCalType, portList);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

autoCalType

Type: **System.Int32**

The type AutoCal Module calibration:

[Autocal1port](#)

[Autocal2port](#)

[AutocalOnePath2port](#)

[Autocal3port](#)

[Autocal4port](#)

portList

Type: **System.Int32 []**

The list of ports.

Return Value

Success or failure code.

Back to [Methods](#)

ExecuteAutoOrientation

Description

Executes the Auto-Orientation procedure of the AutoCal Module. The AutoCal Module must be connected to the ports of Analyzer.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function ExecuteAutoOrientation As Integer
```

C#

```
public int ExecuteAutoOrientation()
```

Visual C++

```
public:  
    int ExecuteAutoOrientation()
```

JavaScript

```
function executeAutoOrientation();
```

Return Value

Success or failure code.

Back to [Methods](#)

ExportLossTable

Description

Saves the loss compensation table into a file.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function ExportLossTable ( _  
    repeatedCapabilitiesID As String, _  
    fileName As String _  
) As Integer
```

C#

```
public int ExportLossTable(  
    string repeatedCapabilitiesID,  
    string fileName  
)
```

Visual C++

```
public:  
    int ExportLossTable(  
        String repeatedCapabilitiesID,  
        String fileName  
    )
```

JavaScript

```
function exportLossTable(repeatedCapabilitiesID, fileName);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Port index i.e. "Port1" and so on.

fileName

Type: **System.String**

The file path into which to save the data. If the full path of the file is not specified, the \CalKit subdirectory of the application directory will be searched for the file. The loss compensation file has *.lct extension by default.

Return Value

Success or failure code.

Back to [Methods](#)

Finalize

Description

Closes the I/O session to the instrument.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Protected Overrides Sub Finalize
```

C#

```
protected override void Finalize()
```

Visual C++

```
protected:  
virtual void Finalize() override
```

JavaScript

```
function finalize();
```

Return Value

Success or failure code.

Back to [Methods](#)

FunctionExecute

Description

Executes the analysis specified for FunctionType integer Property. See more details: Properties.FunctionType.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function FunctionExecute ( _  
    repeatedCapabilitiesID As String _  
) As Integer
```

C#

```
public int FunctionExecute(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
    int FunctionExecute(  
        String repeatedCapabilitiesID  
    )
```

JavaScript

```
function functionExecute(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

Return Value

Success or failure code.

Back to [Methods](#)

GetAttributeViBoolean

Description

Queries the value of a ViBoolean attribute.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetAttributeViBoolean ( _  
    repeatedCapabilitiesID As String, _  
    attributeID As Integer, _  
    <OutAttribute> ByRef value As Boolean _  
    ) As Integer
```

C#

```
public int GetAttributeViBoolean(  
    string repeatedCapabilitiesID,  
    int attributeID,  
    out bool value  
)
```

Visual C++

```
public:  
int GetAttributeViBoolean(  
    String repeatedCapabilitiesID,  
    int attributeID,  
    bool *value  
)
```

JavaScript

```
function getAttributeViBoolean(repeatedCapabilitiesID, attributeID, value);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The repeated capability identifier

attributeID

Type: **System.Int32**

Pass the ID of an attribute. See static class Properties for attribute values.

value

Type: **System.Boolean**

Returns the current value of the attribute.

Return Value

Success or failure code.

Back to [Methods](#)

GetAttributeVInt32

Description

Queries the value of a VInt32 attribute.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetAttributeVInt32 ( _  
    repeatedCapabilitiesID As String, _  
    attributeID As Integer, _  
    <OutAttribute> ByRef value As Integer _  
) As Integer
```

C#

```
public int GetAttributeVInt32(  
    string repeatedCapabilitiesID,  
    int attributeID,  
    out int value  
)
```

Visual C++

```
public:  
int GetAttributeVInt32(  
    String repeatedCapabilitiesID,  
    int attributeID,  
    int *value  
)
```


JavaScript

```
function getAttributeViInt32(repeatedCapabilitiesID, attributeID, value);
```

Parameters

repeatedCapabilitiesID

type: **System.String**

the repeated capability identifier

attributeID

type: **System.Int32**

pass the id of an attribute. see static class properties for attribute values.

value

type: **System.Int32**

returns the current value of the attribute.

Return Value

Success or failure code.

Back to [Methods](#)

GetAttributeViReal64

Description

Queries the value of a ViReal64 attribute.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetAttributeViReal64 ( _  
    repeatedCapabilitiesID As String, _  
    attributeID As Integer, _  
    <OutAttribute> ByRef value As Double _  
    ) As Integer
```

C#

```
public int GetAttributeViReal64(  
    string repeatedCapabilitiesID,  
    int attributeID,  
    out double value  
)
```

Visual C++

```
public:  
int GetAttributeViReal64(  
    String repeatedCapabilitiesID,  
    int attributeID,  
    double *value  
)
```

JavaScript

```
function getAttributeViReal64(repeatedCapabilitiesID, attributeID, value);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The repeated capability identifier

attributeID

Type: **System.Int32**

Pass the ID of an attribute. See static class Properties for attribute values.

value

Type: **System.Double**

Returns the current value of the attribute.

Return Value

Success or failure code.

Back to [Methods](#)

GetAttributeViString

Description

Queries the value of a Vistring attribute.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetAttributeVistring ( _  
    repeatedCapabilitiesID As String, _  
    attributeID As Integer, _  
    ByRef value As String _  
) As Integer
```

C#

```
public int GetAttributeViString(  
    string repeatedCapabilitiesID,  
    int attributeID,  
    ref string value  
)
```

Visual C++

```
public:  
int GetAttributeViString(  
    String repeatedCapabilitiesID,  
    int attributeID,  
    String *value  
)
```

JavaScript

```
function getAttributeViString(repeatedCapabilitiesID, attributeID, value);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The repeated capability identifier

attributeID

Type: **System.Int32**

Pass the ID of an attribute. See static class Properties for attribute values.

value

Type: **System.String**

Returns the current value of the attribute.

Return Value

Success or failure code.

Back to [Methods](#)

GetCalculatedFunctionData

Description

Gets the data array, which is the FunctionExecute function analysis result.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetCalculatedFunctionData ( _  
    repeatedCapabilitiesID As String, _  
    responseValues As Double(), _  
    stimulusValues As Double(), _  
    <OutAttribute> ByRef returnSize As Integer _  
    ) As Integer
```

C#

```
public int GetCalculatedFunctionData(  
    string repeatedCapabilitiesID,  
    double[] responseValues,  
    double[] stimulusValues,  
    out int returnSize  
)
```

Visual C++

```
public:  
int GetCalculatedFunctionData(  
    String repeatedCapabilitiesID,  
    double responseValues[],  
    double stimulusValues[],  
    int *returnSize
```

Visual C++

```
)
```

JavaScript

```
function getCalculatedFunctionData(repeatedCapabilitiesID,  
responseValues, stimulusValues, returnSize);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

responseValues

Type: **System.Double** []

The array of the response value.

stimulusValues

Type: **System.Double** []

The array of the stimulus value. Always set to 0 for the analysis of mean value, standard deviation, and peak–to–peak value.

returnSize

Type: **System.Int32**

The actual number of elements filled by the query.

Return Value

Success or failure code.

Back to [Methods](#)

GetCalibrationInfo

Description

Gets the information string of the calibration acting between the source and receiver ports.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetCalibrationInfo ( _  
    repeatedCapabilitiesID As String, _  
    receiverPort As Integer, _  
    sourcePort As Integer, _  
    <OutAttribute> ByRef value As String _  
    ) As Integer
```

C#

```
public int GetCalibrationInfo(  
    string repeatedCapabilitiesID,  
    int receiverPort,  
    int sourcePort,  
    out string value  
)
```

Visual C++

```
public:  
int GetCalibrationInfo(  
    String repeatedCapabilitiesID,  
    int receiverPort,  
    int sourcePort,
```


Visual C++

```
String value[]  
)
```

JavaScript

```
function getCalibrationInfo(repeatedCapabilitiesID, receiverPort, sourcePort,  
value);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

receiverPort

Type: **System.Int32**

The number of port from which the measurement is performed.

sourcePort

Type: **System.Int32**

The number of port which is the source.

value

Type: **System.String**

The information [string](#) of the calibration acting between the source and receiver ports.

Return Value

Success or failure code.

Back to [Methods](#)

GetCalKitOrderLoad

Description

Gets the number of the calibration standard of the load type, used for the measurement of the specified port.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetCalKitOrderLoad ( _  
    port As Integer, _  
    <OutAttribute> ByRef stdNumber As Integer _  
) As Integer
```

C#

```
public int GetCalKitOrderLoad(  
    int port,  
    out int stdNumber  
)
```

Visual C++

```
public:  
int GetCalKitOrderLoad(  
    int port,  
    int *stdNumber  
)
```

JavaScript

```
function getCalKitOrderLoad(port, stdNumber);
```

Parameters

port

Type: **System.Int32**

The number of port from which the measurement is performed.

stdNumber

Type: **System.Int32**

The number of the calibration standard.

Return Value

Success or failure code.

Back to [Methods](#)

GetCalKitOrderOpen

Description

Gets the number of the calibration standard of the open type, used for the measurement of the specified port.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetCalKitOrderOpen ( _  
    port As Integer, _  
    <OutAttribute> ByRef stdNumber As Integer _  
    ) As Integer
```

C#

```
public int GetCalKitOrderOpen(  
    int port,  
    out int stdNumber  
)
```

Visual C++

```
public:  
int GetCalKitOrderOpen(  
    int port,  
    int *stdNumber  
)
```

JavaScript

```
function getCalKitOrderOpen(port, stdNumber);
```

Parameters

port

Type: **System.Int32**

The number of port from which the measurement is performed.

stdNumber

Type: **System.Int32**

The number of the calibration standard.

Return Value

Success or failure code.

Back to [Methods](#)

GetCalKitOrderShort

Description

Gets the number of the calibration standard of the short type, used for the measurement of the specified port.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetCalKitOrderShort ( _  
    port As Integer, _  
    <OutAttribute> ByRef stdNumber As Integer _  
) As Integer
```

C#

```
public int GetCalKitOrderShort(  
    int port,  
    out int stdNumber  
)
```

Visual C++

```
public:  
int GetCalKitOrderShort(  
    int port,  
    int *stdNumber  
)
```

JavaScript

```
function getCalKitOrderShort(port, stdNumber);
```

Parameters

port

Type: **System.Int32**

The number of port from which the measurement is performed.

stdNumber

Type: **System.Int32**

The number of the calibration standard.

Return Value

Success or failure code.

Back to [Methods](#)

GetCalKitOrderThru

Description

Gets the number of the calibration standard of the thru type, used for the measurement between the source and receiver ports.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetCalKitOrderThru ( _  
    receiverPort As Integer, _  
    sourcePort As Integer, _  
    <OutAttribute> ByRef stdNumber As Integer _  
) As Integer
```

C#

```
public int GetCalKitOrderThru(  
    int receiverPort,  
    int sourcePort,  
    out int stdNumber  
)
```

Visual C++

```
public:  
int GetCalKitOrderThru(  
    int receiverPort,  
    int sourcePort,  
    int *stdNumber  
)
```


JavaScript

```
function getCalKitOrderThru(receiverPort, sourcePort, stdNumber);
```

Parameters

receiverPort

Type: **System.Int32**

The number of port from which the measurement is performed.

sourcePort

Type: **System.Int32**

The number of port which is the source.

stdNumber

Type: **System.Int32**

The number of the calibration standard.

Return Value

Success or failure code.

Back to [Methods](#)

GetCalKitOrderTrlLine

Description

Gets the number of the calibration standard of the TRL line type, used for the measurement between the source and receiver ports.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetCalKitOrderTrlLine ( _  
    receiverPort As Integer, _  
    sourcePort As Integer, _  
    <OutAttribute> ByRef stdNumber As Integer _  
) As Integer
```

C#

```
public int GetCalKitOrderTrlLine(  
    int receiverPort,  
    int sourcePort,  
    out int stdNumber  
)
```

Visual C++

```
public:  
int GetCalKitOrderTrlLine(  
    int receiverPort,  
    int sourcePort,  
    int *stdNumber  
)
```

JavaScript

```
function getCalKitOrderTrlLine(receiverPort, sourcePort, stdNumber);
```

Parameters

receiverPort

Type: **System.Int32**

The number of port from which the measurement is performed.

sourcePort

Type: **System.Int32**

The number of port which is the source.

stdNumber

Type: **System.Int32**

The number of the calibration standard.

Return Value

Success or failure code.

Back to [Methods](#)

GetCalKitOrderTrlReflect

Description

Gets the number of the calibration standard of the TRL Reflect type, used for the measurement of the specified port.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetCalKitOrderTrlReflect ( _  
    port As Integer, _  
    <OutAttribute> ByRef stdNumber As Integer _  
    ) As Integer
```

C#

```
public int GetCalKitOrderTrlReflect(  
    int port,  
    out int stdNumber  
)
```

Visual C++

```
public:  
int GetCalKitOrderTrlReflect(  
    int port,  
    int *stdNumber  
)
```

JavaScript

```
function getCalKitOrderTrlReflect(port, stdNumber);
```

Parameters

port

Type: **System.Int32**

The number of port from which the measurement is performed.

stdNumber

Type: **System.Int32**

The number of the calibration standard.

Return Value

Success or failure code.

Back to [Methods](#)

GetCalKitOrderTrlThru

Description

Gets the number of the calibration standard of the TRL thru type, used for the measurement between the source and receiver ports.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetCalKitOrderTrlThru ( _  
    receiverPort As Integer, _  
    sourcePort As Integer, _  
    <OutAttribute> ByRef stdNumber As Integer _  
) As Integer
```

C#

```
public int GetCalKitOrderTrlThru(  
    int receiverPort,  
    int sourcePort,  
    out int stdNumber  
)
```

Visual C++

```
public:  
int GetCalKitOrderTrlThru(  
    int receiverPort,  
    int sourcePort,  
    int *stdNumber  
)
```

JavaScript

```
function getCalKitOrderTrlThru(receiverPort, sourcePort, stdNumber);
```

Parameters

receiverPort

Type: **System.Int32**

The number of port from which the measurement is performed.

sourcePort

Type: **System.Int32**

The number of port which is the source.

stdNumber

Type: **System.Int32**

The number of the calibration standard.

Return Value

Success or failure code.

Back to [Methods](#)

GetCalStandardS1PData

Description

Gets the data array of the data-based calibration standard. The first element of the array is 1 and determines the number of ports of the calibration standard.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetCalStandardS1PData ( _  
    repeatedCapabilitiesID As String, _  
    stimulusValues As Double(), _  
    realS11Values As Double(), _  
    imageS11Values As Double(), _  
    <OutAttribute> ByRef returnSize As Integer _  
    ) As Integer
```

C#

```
public int GetCalStandardS1PData(  
    string repeatedCapabilitiesID,  
    double[] stimulusValues,  
    double[] realS11Values,  
    double[] imageS11Values,  
    out int returnSize  
)
```

Visual C++

```
public:  
int GetCalStandardS1PData(  
    String repeatedCapabilitiesID[],
```


Visual C++

```
double stimulusValues[],  
double realS11Values[],  
double imageS11Values[],  
int *returnSize  
)
```

JavaScript

```
function getCalStandardS1PData(repeatedCapabilitiesID, stimulusValues,  
realS11Values, imageS11Values, returnSize);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.

stimulusValues

Type: **System.Double** []

The array of stimulus value.

realS11Values

Type: **System.Double** []

The array of value real part S11.

imageS11Values

Type: **System.Double** []

The array of value image part S11.

returnSize

Type: **System.Int32**

The actual number of elements filled by the query.

Return Value

Success or failure code.

Back to [Methods](#)

GetCalStandardS2PData

Description

Gets the data array of the data-based calibration standard. The first element of the array is 2 and determines the number of ports of the calibration standard.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetCalStandardS2PData ( _  
    repeatedCapabilitiesID As String, _  
    stimulusValues As Double(), _  
    realS11Values As Double(), _  
    imageS11Values As Double(), _  
    realS21Values As Double(), _  
    imageS21Values As Double(), _  
    realS12Values As Double(), _  
    imageS12Values As Double(), _  
    realS22Values As Double(), _  
    imageS22Values As Double(), _  
    <OutAttribute> ByRef returnSize As Integer _  
    ) As Integer
```

C#

```
public int GetCalStandardS2PData(  
    string repeatedCapabilitiesID,  
    double[] stimulusValues,  
    double[] realS11Values,  
    double[] imageS11Values,  
    double[] realS21Values,
```

C#

```
double[] imageS21Values,  
double[] realS12Values,  
double[] imageS12Values,  
double[] realS22Values,  
double[] imageS22Values,  
out int returnSize  
)
```

Visual C++

```
public:  
int GetCalStandardS2PData(  
    String repeatedCapabilitiesID,  
    double stimulusValues[],  
    double realS11Values[],  
    double imageS11Values[],  
    double realS21Values[],  
    double imageS21Values[],  
    double realS12Values[],  
    double imageS12Values[],  
    double realS22Values[],  
    double imageS22Values[],  
    int *returnSize  
)
```

JavaScript

```
function getCalStandardS2PData(repeatedCapabilitiesID, stimulusValues,  
realS11Values, imageS11Values, realS21Values, imageS21Values,  
realS12Values, imageS12Values, realS22Values, imageS22Values,  
returnSize);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.

stimulusValues

Type: **System.Double** []

The array of stimulus value.

realS11Values

Type: **System.Double** []

The array of value real part S11.

imageS11Values

Type: **System.Double** []

The array of value image part S11.

realS21Values

Type: **System.Double** []

The array of value real part S21.

imageS21Values

Type: **System.Double** []

The array of value image part S21.

realS12Values

Type: **System.Double** []

The array of value real part S12.

imageS12Values

Type: **System.Double** []

The array of value image part S12.

realS22Values

Type: **System.Double** []

The array of value real part S22.

imageS22Values

Type: **System.Double** []

The array of value image part S22.

returnSize

Type: **System.Int32**

The actual number of elements filled by the query.

Return Value

Success or failure code.

Back to [Methods](#)

GetError

Description

This function retrieves and then clears the VI error information for the session or the current execution thread.

If the user specifies a valid VI session for the Vi parameter, GetError retrieves and then clears the error information for the session.

If the user passes VI_NULL for the Vi parameter, GetError retrieves and then clears the error information for the current execution thread.

If the Vi parameter is an invalid session, the function does nothing and returns an error.

Normally, the error information describes the first error that occurred since the user last called GetError or ClearError function.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetError ( _  
    <OutAttribute> ByRef errorCode As Integer, _  
    <OutAttribute> ByRef description As String _  
) As Integer
```

C#

```
public int GetError(  
    out int errorCode,  
    out string description  
)
```

Visual C++

```
public:
```

Visual C++

```
int GetError(  
    int *errorCode,  
    String description[]  
)
```

JavaScript

```
function getError(errorCode, description);
```

Parameters

errorCode

Type: **System.Int32**

Passes the `returnValue` parameter that is returned from any driver function. The default value is 0 (VI_SUCCESS).

description

Type: **System.String**

Returns the error description for the session or execution thread.

If there is no description, the `GetError` function returns an empty string.

The buffer must contain at least as many elements as the value you specify with `bufferSize`.

If the error description, including the terminating NULL byte, contains more bytes than you indicate with `bufferSize`, the function copies `bufferSize - 1` bytes into the buffer, places an ASCII NULL byte at the end of the buffer, and returns the size of the buffer that you must pass to get the entire value.

If you pass 0 to `bufferSize`, you can pass `VI_NULL` for this parameter.

Return Value

Success or failure code.

Back to [Methods](#)

GetLimitTestData

Description

Gets the data array, which is the limit line in the limit test function.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetLimitTestData ( _  
    repeatedCapabilitiesID As String, _  
    <OutAttribute> ByRef segmentCount As Integer, _  
    limitLineType As Integer(), _  
    beginLimitArgument As Double(), _  
    endLimitArgument As Double(), _  
    beginLimitValues As Double(), _  
    endLimitValues As Double() _  
) As Integer
```

C#

```
public int GetLimitTestData(  
    string repeatedCapabilitiesID,  
    out int segmentCount,  
    int[] limitLineType,  
    double[] beginLimitArgument,  
    double[] endLimitArgument,  
    double[] beginLimitValues,  
    double[] endLimitValues  
)
```

Visual C++

```
public:
int GetLimitTestData(
    String repeatedCapabilitiesID,
    int *segmentCount,
    int limitLineType[],
    double beginLimitArgument[],
    double endLimitArgument[],
    double beginLimitValues[],
    double endLimitValues[]
)
```

JavaScript

```
function getLimitTestData(repeatedCapabilitiesID, segmentCount,
limitLineType, beginLimitArgument, endLimitArgument, beginLimitValues,
endLimitValues);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

segmentCount

Type: **System.Int32**

The number of limit line segments N is from 0 to 100.

limitLineType

Type: **System.Int32** []

The type of the n–th limit line segment.

[AnalysisLimitLineOff](#)

[AnalysisLimitLineMax](#)

[AnalysisLimitLineMin](#)

[AnalysisLimitLineSingle](#)

beginLimitArgument

Type: **System.Double** []

The stimulus value in the start [point](#) of the n–th segment.

endLimitArgument

Type: **System.Double** []

The stimulus value in the end [point](#) of the n–th segment.

beginLimitValues

Type: **System.Double** []

The response value in the start [point](#) of the n–th segment.

endLimitValues

Type: **System.Double** []

The response value in the end [point](#) of the n–th segment.

Return Value

Success or failure code.

Back to [Methods](#)

GetLimitTestReportAll

Description

Gets the data array, which is the limit test result.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetLimitTestReportAll ( _  
    repeatedCapabilitiesID As String, _  
    stimulusValues As Double(), _  
    resultValues As Integer(), _  
    upperLimitValues As Double(), _  
    lowerLimitValues As Double() _  
    ) As Integer
```

C#

```
public int GetLimitTestReportAll(  
    string repeatedCapabilitiesID,  
    double[] stimulusValues,  
    int[] resultValues,  
    double[] upperLimitValues,  
    double[] lowerLimitValues  
)
```

Visual C++

```
public:  
int GetLimitTestReportAll(  
    String repeatedCapabilitiesID,  
    double stimulusValues[],
```

Visual C++

```
int resultValues[],  
double upperLimitValues[],  
double lowerLimitValues[]  
)
```

JavaScript

```
function getLimitTestReportAll(repeatedCapabilitiesID, stimulusValues,  
resultValues, upperLimitValues, lowerLimitValues);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

stimulusValues

Type: **System.Double** []

The stimulus value in the n–th point.

resultValues

Type: **System.Int32** []

The limit test result in the n–th point. –1: No limit, 0: Fail, 1: Pass

upperLimitValues

Type: **System.Double** []

The upper limit value in the n–th point (0 – if there is no limit).

lowerLimitValues

Type: **System.Double** []

The lower limit value in the n–th point (0 – if there is no limit).

Return Value

Success or failure code.

Back to [Methods](#)

GetLimitTestReport

Description

Gets the data array, which is the stimulus values of the measurement points that failed the limit test.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetLimitTestReport ( _  
    repeatedCapabilitiesID As String, _  
    <OutAttribute> ByRef points As Integer, _  
    dataValues As Double() _  
) As Integer
```

C#

```
public int GetLimitTestReport(  
    string repeatedCapabilitiesID,  
    out int points,  
    double[] dataValues  
)
```

Visual C++

```
public:  
int GetLimitTestReport(  
    String repeatedCapabilitiesID,  
    int *points,  
    double dataValues[]  
)
```

JavaScript

```
function getLimitTestReport(repeatedCapabilitiesID, points, dataValues);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

points

Type: **System.Int32**

The number of the measurement points that failed the limit test.

dataValues

Type: **System.Double** []

The stimulus values of the measurement *points* that failed the limit test.

Return Value

Success or failure code.

Back to [Methods](#)

GetLimitTestStatus

Description

Gets the limit test result.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetLimitTestStatus ( _  
    repeatedCapabilitiesID As String, _  
    <OutAttribute> ByRef result As Boolean _  
    ) As Integer
```

C#

```
public int GetLimitTestStatus(  
    string repeatedCapabilitiesID,  
    out bool result  
)
```

Visual C++

```
public:  
int GetLimitTestStatus(  
    String repeatedCapabilitiesID,  
    bool *result  
)
```

JavaScript

```
function getLimitTestStatus(repeatedCapabilitiesID, result);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

result

Type: **System.Boolean**

The limit test result.

Return Value

Success or failure code.

Back to [Methods](#)

GetMarkerValue

Description

Gets the response value of the marker. If the reference marker is turned ON, the values of the markers from 1 to 15 are read **out** as relative values to the reference marker.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetMarkerValue ( _  
    repeatedCapabilitiesID As String, _  
    <OutAttribute> ByRef realPart As Double, _  
    <OutAttribute> ByRef imagePart As Double _  
    ) As Integer
```

C#

```
public int GetMarkerValue(  
    string repeatedCapabilitiesID,  
    out double realPart,  
    out double imagePart  
)
```

Visual C++

```
public:  
int GetMarkerValue(  
    String repeatedCapabilitiesID,  
    double *realPart,  
    double *imagePart  
)
```

JavaScript

```
function getMarkerValue(repeatedCapabilitiesID, realPart, imagePart);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index and Marker index for the active trace of channel i.e. "Channel1:Marker2" and so on. The physical names are supported along with the corresponding Channel index and Measurement index and Marker index for the n–th trace of channel i.e. "Channel1:Measurement3:Marker2" and so on.

realPart

Type: **System.Double**

The real number in rectangular format, real part in polar and Smith chart formats.

imagePart

Type: **System.Double**

0 in rectangular format, imaginary part in polar and Smith chart formats.

Return Value

Success or failure code.

Back to [Methods](#)

GetMaxChannelCount

Description

Gets the maximum number of channels.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetMaxChannelCount ( _  
    <OutAttribute> ByRef channelCount As Integer _  
) As Integer
```

C#

```
public int GetMaxChannelCount(  
    out int channelCount  
)
```

Visual C++

```
public:  
int GetMaxChannelCount(  
    int *channelCount  
)
```

JavaScript

```
function getMaxChannelCount(channelCount);
```

Parameters

channelCount

Type: **System.Int32**

The maximum number of channels.

Return Value

Success or failure code.

Back to [Methods](#)

GetMaxTraceCount

Description

Gets the maximum number of traces in the channel.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetMaxTraceCount ( _  
    <OutAttribute> ByRef traceCount As Integer _  
) As Integer
```

C#

```
public int GetMaxTraceCount(  
    out int traceCount  
)
```

Visual C++

```
public:  
int GetMaxTraceCount(  
    int *traceCount  
)
```

JavaScript

```
function getMaxTraceCount(traceCount);
```

Parameters

traceCount

Type: **System.Int32**

The maximum number of traces in the channel.

Return Value

Success or failure code.

Back to [Methods](#)

GetPowerCalibrationTable

Description

Gets the power correction array (result of power calibration executed by PerformCalibrationAction function).

The array size is NOP, where NOP is the number of measurement points.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetPowerCalibrationTable ( _  
    repeatedCapabilitiesID As String, _  
    powerValues As Double(), _  
    <OutAttribute> ByRef returnSize As Integer _  
    ) As Integer
```

C#

```
public int GetPowerCalibrationTable(  
    string repeatedCapabilitiesID,  
    double[] powerValues,  
    out int returnSize  
)
```

Visual C++

```
public:  
int GetPowerCalibrationTable(  
    String repeatedCapabilitiesID,  
    double powerValues[],  
    int *returnSize  
)
```

JavaScript

```
function getPowerCalibrationTable(repeatedCapabilitiesID, powerValues,  
returnSize);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.

powerValues

Type: **System.Double[]**

The array of the power correction value.

returnSize

Type: **System.Int32**

The actual number of elements filled by the query.

Return Value

Success or failure code.

Back to [Methods](#)

GetPowerLossCompensationTable

Description

Gets the loss compensation table used when the power calibration is executed by PerformCalibrationAction function.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetPowerLossCompensationTable ( _  
    repeatedCapabilitiesID As String, _  
    frequencyValues As Double(), _  
    lossValues As Double(), _  
    <OutAttribute> ByRef returnSize As Integer _  
    ) As Integer
```

C#

```
public int GetPowerLossCompensationTable(  
    string repeatedCapabilitiesID,  
    double[] frequencyValues,  
    double[] lossValues,  
    out int returnSize  
)
```

Visual C++

```
public:  
int GetPowerLossCompensationTable(  
    String repeatedCapabilitiesID,  
    double frequencyValues[],  
    double lossValues[],
```

Visual C++

```
int *returnSize  
)
```

JavaScript

```
function      getPowerLossCompensationTable(repeatedCapabilitiesID,  
frequencyValues, lossValues, returnSize);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.

frequencyValues

Type: **System.Double** []

The array of value of frequency.

lossValues

Type: **System.Double** []

The array of value of the loss compensation (from –100 to +100 dB).

returnSize

Type: **System.Int32**

The actual number of elements filled by the query.

Return Value

Success or failure code.

Back to [Methods](#)

GetRippleLimitData

Description

Gets the data array, which is the limit line for the ripple limit function.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetRippleLimitData ( _  
    repeatedCapabilitiesID As String, _  
    <OutAttribute> ByRef segmentCount As Integer, _  
    rippleLimitType As Integer(), _  
    beginLimitArgument As Double(), _  
    endLimitArgument As Double(), _  
    limitValues As Double() _  
    ) As Integer
```

C#

```
public int GetRippleLimitData(  
    string repeatedCapabilitiesID,  
    out int segmentCount,  
    int[] rippleLimitType,  
    double[] beginLimitArgument,  
    double[] endLimitArgument,  
    double[] limitValues  
)
```

Visual C++

```
public:  
    int GetRippleLimitData(  

```

Visual C++

```
String repeatedCapabilitiesID,  
int *segmentCount,  
int rippleLimitType[],  
double beginLimitArgument[],  
double endLimitArgument[],  
double limitValues[]  
)
```

JavaScript

```
function getRippleLimitData(repeatedCapabilitiesID, segmentCount,  
rippleLimitType, beginLimitArgument, endLimitArgument, limitValues);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

segmentCount

Type: **System.Int32**

The number of limit line segments N is the integer from 0 to 12. Setting 0 clears the limit line.

rippleLimitType

Type: **System.Int32 []**

The type of the n–th limit line segment.

[AnalysisRippleLimitOff](#)

[AnalysisRippleLimitOn](#)

beginLimitArgument

Type: **System.Double** []

The stimulus value in the beginning [point](#) of the n–th segment.

endLimitArgument

Type: **System.Double** []

The stimulus value in the end [point](#) of the n–th segment.

limitValues

Type: **System.Double** []

The ripple limit value of the n–th segment.

Return Value

Success or failure code.

Back to [Methods](#)

GetRippleLimitTestReport

Description

Gets the data array, which is the ripple limit test result.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetRippleLimitTestReport ( _  
    repeatedCapabilitiesID As String, _  
    <OutAttribute> ByRef bandsCount As Integer, _  
    bandNumbers As Integer(), _  
    bandRippleValues As Double(), _  
    bandResults As Integer() _  
    ) As Integer
```

C#

```
public int GetRippleLimitTestReport(  
    string repeatedCapabilitiesID,  
    out int bandsCount,  
    int[] bandNumbers,  
    double[] bandRippleValues,  
    int[] bandResults  
)
```

Visual C++

```
public:  
int GetRippleLimitTestReport(  
    String repeatedCapabilitiesID,  
    int *bandsCount,
```


Visual C++

```
int bandNumbers[],  
double bandRippleValues[],  
int bandResults[]  
)
```

JavaScript

```
function getRippleLimitTestReport(repeatedCapabilitiesID, bandsCount,  
bandNumbers, bandRippleValues, bandResults);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

bandsCount

Type: **System.Int32**

N total number of the bands.

bandNumbers

Type: **System.Int32** []

n number of the band

bandRippleValues

Type: **System.Double** []

The ripple value in the n–th band.

bandResults

Type: **System.Int32** []

The ripple limit test result in the n -th band: 0- Pass, 1- Fail.

Return Value

Success or failure code.

Back to [Methods](#)

GetRippleLimitTestStatus

Description

Gets the ripple limit test result.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetRippleLimitTestStatus ( _  
    repeatedCapabilitiesID As String, _  
    <OutAttribute> ByRef result As Boolean _  
    ) As Integer
```

C#

```
public int GetRippleLimitTestStatus(  
    string repeatedCapabilitiesID,  
    out bool result  
)
```

Visual C++

```
public:  
int GetRippleLimitTestStatus(  
    String repeatedCapabilitiesID,  
    bool *result  
)
```

JavaScript

```
function getRippleLimitTestStatus(repeatedCapabilitiesID, result);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

result

Type: **System.Boolean**

The ripple limit test result.

Return Value

Success or failure code.

Back to [Methods](#)

GetSegmentData

Description

Gets the array of the segment sweep table.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetSegmentData ( _  
    repeatedCapabilitiesID As String, _  
    <OutAttribute> ByRef segmentCount As Integer, _  
    startFrequencies As Double(), _  
    stopFrequencies As Double(), _  
    numberPoints As Integer(), _  
    <OutAttribute> ByRef listIFBandwidth As Boolean, _  
    ifBandwidthValues As Double(), _  
    <OutAttribute> ByRef listPower As Boolean, _  
    powerValues As Double(), _  
    <OutAttribute> ByRef listDelay As Boolean, _  
    delayValues As Double(), _  
    <OutAttribute> ByRef listTime As Boolean, _  
    timeValues As Double() _  
) As Integer
```

C#

```
public int GetSegmentData(  
    string repeatedCapabilitiesID,  
    out int segmentCount,  
    double[] startFrequencies,  
    double[] stopFrequencies,
```

C#

```
    int[] numberPoints,  
    out bool listIFBandwidth,  
    double[] ifBandwidthValues,  
    out bool listPower,  
    double[] powerValues,  
    out bool listDelay,  
    double[] delayValues,  
    out bool listTime,  
    double[] timeValues  
)
```

Visual C++

```
public:  
int GetSegmentData(  
    String repeatedCapabilitiesID,  
    int *segmentCount,  
    double startFrequencies[],  
    double stopFrequencies[],  
    int numberPoints[],  
    bool *listIFBandwidth,  
    double ifBandwidthValues[],  
    bool *listPower,  
    double powerValues[],  
    bool *listDelay,  
    double delayValues[],  
    bool *listTime,  
    double* timeValues[]  
)
```

JavaScript

```
function getSegmentData(repeatedCapabilitiesID, segmentCount,  
startFrequencies, stopFrequencies, numberPoints, listIFBandwidth,  
ifBandwidthValues, listPower, powerValues, listDelay, delayValues, listTime,  
timeValues);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

segmentCount

Type: **System.Int32**

The number of segments.

startFrequencies

Type: **System.Double** []

The array of start values.

stopFrequencies

Type: **System.Double** []

The array of stop values.

numberPoints

Type: **System.Int32** []

The array of number of points.

listIFBandwidth

Type: **System.Boolean**

Setting of *ifbwValues* (0 – disabled, 1 – enabled).

ifBandwidthValues

Type: **System.Double** []

The array of value of IF bandwidth (if enabled).

listPower

Type: **System.Boolean**

Setting of *powerValues* (0 – disabled, 1 – enabled).

powerValues

Type: **System.Double** []

The array of value of power (if enabled).

listDelay

Type: **System.Boolean**

Setting of *delayValues* (0 – disabled, 1 – enabled).

delayValues

Type: **System.Double** []

The array of value of measurement delay (if enabled).

listTime

Type: **System.Boolean**

Setting of *timeValues* (0 – disabled, 1 – enabled).

timeValues

Type: **System.Double** []

Reserved for future use (if enabled).

Return Value

Success or failure code.

Back to [Methods](#)

GetSubclassStdOrder

Description

Gets the subclass used to specify classes of calibration standards by functions:

[GetCalKitOrderOpen](#)

[GetCalKitOrderShort](#)

[GetCalKitOrderLoad](#)

[GetCalKitOrderThru](#)

[GetCalKitOrderTrlLine](#)

[GetCalKitOrderTrlThru](#)

[GetCalKitOrderTrlReflect](#)

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetSubclassStdOrder ( _  
    <OutAttribute> ByRef subclassNumber As Integer _  
    ) As Integer
```

C#

```
public int GetSubclassStdOrder(  
    out int subclassNumber  
)
```

Visual C++

```
public:  
int GetSubclassStdOrder(  
    int *subclassNumber
```

Visual C++

```
)
```

JavaScript

```
function getSubclassStdOrder(subclassNumber);
```

Parameters

subclassNumber

Type: **System.Int32**

The subclass number of the calibration standard.

Return Value

Success or failure code.

Back to [Methods](#)

GetTouchstoneFileType

Description

Gets the Touchstone file type and the port numbers, when saving S-parameters by [SaveTouchstoneFile](#) function.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function GetTouchstoneFileType ( _  
    repeatedCapabilitiesID As String, _  
    <OutAttribute> ByRef touchstoneType As Integer, _  
    bufferSize As Integer, _  
    ports As Integer() _  
    ) As Integer
```

C#

```
public int GetTouchstoneFileType(  
    string repeatedCapabilitiesID,  
    out int touchstoneType,  
    int bufferSize,  
    int[] ports  
)
```

Visual C++

```
public:  
int GetTouchstoneFileType(  
    String repeatedCapabilitiesID,  
    int *touchstoneType,  
    int bufferSize,
```

Visual C++

```
    int ports[]  
)
```

JavaScript

```
function getTouchstoneFileType(repeatedCapabilitiesID, touchstoneType,  
bufferSize, ports);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

touchstoneType

Type: **System.Int32**

The type of Touchstone file:

[SaveTouchstoneFileS1p](#)

[SaveTouchstoneFileS2p](#)

[SaveTouchstoneFileS3p](#)

[SaveTouchstoneFileS4p](#)

bufferSize

Type: **System.Int32**

The size of the arrays to hold in the ports parameter.

ports

Type: **System.Int32 []**

The array of port numbers.

Return Value

Success or failure code.

Back to [Methods](#)

HoldAllChannels

Description

Turns OFF the continuous trigger initiation mode for all channels.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function HoldAllChannels ( _  
    repeatedCapabilitiesID As String _  
) As Integer
```

C#

```
public int HoldAllChannels(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int HoldAllChannels(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function holdAllChannels(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

Pass VI_NULL or empty string if operation does not apply to a repeated capability.

Return Value

Success or failure code.

Back to [Methods](#)

ImportLossTable

Description

Recalls the loss compensation file. The file must be saved by the ExportLossTable function.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function ImportLossTable ( _  
    repeatedCapabilitiesID As String, _  
    fileName As String _  
) As Integer
```

C#

```
public int ImportLossTable(  
    string repeatedCapabilitiesID,  
    string fileName  
)
```

Visual C++

```
public:  
int ImportLossTable(  
    String repeatedCapabilitiesID,  
    String fileName  
)
```

JavaScript

```
function importLossTable(repeatedCapabilitiesID, fileName);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Port index i.e. "Port1" and so on.

fileName

Type: **System.String**

The file path into which to save the data. If the full path of the file is not specified, the \CalKit subdirectory of the application directory will be searched for the file. The loss compensation file has *.lct extension by default.

Return Value

Success or failure code.

Back to [Methods](#)

InitNPortUserCal

Description

Initializes N-port user calibration.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function InitNPortUserCal ( _  
    repeatedCapabilitiesID As String, _  
    calibrationType As Integer, _  
    portList As Integer() _  
) As Integer
```

C#

```
public int InitNPortUserCal(  
    string repeatedCapabilitiesID,  
    int calibrationType,  
    int[] portList  
)
```

Visual C++

```
public:  
int InitNPortUserCal(  
    String repeatedCapabilitiesID,  
    int calibrationType,  
    int portList[]  
)
```

JavaScript

```
function initNPortUserCal(repeatedCapabilitiesID, calibrationType, portList);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

calibrationType

Type: **System.Int32**

The type of calibration that is performed. See more details: Properties.[*calibrationType*](#)

portList

Type: **System.Int32 []**

The list of ports.

Return Value

Success or failure code.

Back to [Methods](#)

InitUserCal

Description

Initializes user calibration.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function InitUserCal ( _  
    repeatedCapabilitiesID As String, _  
    calibrationType As Integer, _  
    receiverPort As Integer, _  
    sourcePort As Integer _  
) As Integer
```

C#

```
public int InitUserCal(  
    string repeatedCapabilitiesID,  
    int calibrationType,  
    int receiverPort,  
    int sourcePort  
)
```

Visual C++

```
public:  
int InitUserCal(  
    String repeatedCapabilitiesID,  
    int calibrationType,  
    int receiverPort,  
    int sourcePort
```

Visual C++

```
)
```

JavaScript

```
function initUserCal(repeatedCapabilitiesID, calibrationType, receiverPort,  
sourcePort);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

calibrationType

Type: **System.Int32**

The type of calibration that is performed. See more details: [CalibrationType](#)

receiverPort

Type: **System.Int32**

The number of port from which the measurement is performed.

sourcePort

Type: **System.Int32**

The number of port which is the source.

Return Value

Success or failure code.

Back to [Methods](#)

LoadTouchstoneFile

Description

Loads the Touchstone file with the specified name to the measured S-parameters of the active channel.

The Touchstone file types 1, 2, 3 or 4 port (file extensions s1p, s2p, s3p or s4p) are supported.

On completion of the command, the channel goes to the hold state.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function LoadTouchstoneFile ( _  
    repeatedCapabilitiesID As String, _  
    destination As Integer, _  
    fileName As String _  
) As Integer
```

C#

```
public int LoadTouchstoneFile(  
    string repeatedCapabilitiesID,  
    int destination,  
    string fileName  
)
```

Visual C++

```
public:  
int LoadTouchstoneFile(  
    String repeatedCapabilitiesID,  
    int destination,
```

Visual C++

```
String fileName  
)
```

JavaScript

```
function loadTouchstoneFile(repeatedCapabilitiesID, destination, fileName);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

Must be VI_NULL or an empty string if loads the Touchstone file to the measured S-parameters of the active channel. If loads the Touchstone file with the specified name to the memory trace, the physical names are supported along with the corresponding Measurement index i.e. "Measurement1" and so on.

destination

Type: **System.Int32**

The destination:

[UploadTouchstoneFileToActiveTraceMemory](#)

[UploadTouchstoneFileToSParameters](#)

fileName

Type: **System.String**

The file path into which to save the data. If the full path of the file is not specified, the \FixtureSim subdirectory of the application directory will be searched for the file. The file has *.sNp extension by default.

Return Value

Success or failure code.

Back to [Methods](#)

LockSession

Description

Obtains a multithread lock on the instrument session.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function LockSession ( _  
    <OutAttribute> ByRef callerHasLock As Boolean _  
) As Integer
```

C#

```
public int LockSession(  
    out bool callerHasLock  
)
```

Visual C++

```
public:  
int LockSession(  
    bool *callerHasLock  
)
```

JavaScript

```
function lockSession(callerHasLock);
```

Parameters

callerHasLock

Type: **System.Boolean**

Return Value

Success or failure code.

Back to [Methods](#)

MarkerFunctionExecute

Description

Executes the marker search according to the specified criterion.

The type of the marker search is set by [MarkerSearchType](#).

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function MarkerFunctionExecute ( _  
    repeatedCapabilitiesID As String _  
) As Integer
```

C#

```
public int MarkerFunctionExecute(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int MarkerFunctionExecute(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function markerFunctionExecute(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index and Marker index for the active trace of channel i.e. "Channel1:Marker2" and so on. The physical names are supported along with the corresponding Channel index and Measurement index and Marker index for the n-th trace of channel i.e. "Channel1:Measurement3:Marker2" and so on.

Return Value

Success or failure code.

Back to [Methods](#)

MarkerFunctionsMarkerCenter

Description

Sets the value of the specified item to the value of the position of the marker.

Sweep center value is set to the stimulus value of the marker position.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function MarkerFunctionsMarkerCenter ( _  
    repeatedCapabilitiesID As String _  
    ) As Integer
```

C#

```
public int MarkerFunctionsMarkerCenter(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int MarkerFunctionsMarkerCenter(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function markerFunctionsMarkerCenter(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index and Marker index for the active trace of channel i.e. "Channel1:Marker2" and so on. The physical names are supported along with the corresponding Channel index, Measurement index and Marker index for the n-th trace of channel i.e. "Channel1:Measurement3:Marker2" and so on.

Return Value

Success or failure code.

Back to [Methods](#)

MarkerFunctionsMarkerDelay

Description

Sets the value of the specified item to the value of the position of the marker.

Delay value is set to the response value of the marker position.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function MarkerFunctionsMarkerDelay ( _  
    repeatedCapabilitiesID As String _  
    ) As Integer
```

C#

```
public int MarkerFunctionsMarkerDelay(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int MarkerFunctionsMarkerDelay(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function markerFunctionsMarkerDelay(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index and Marker index for the active trace of channel i.e. "Channel1:Marker2" and so on. The physical names are supported along with the corresponding Channel index, Measurement index and Marker index for the n-th trace of channel i.e. "Channel1:Measurement3:Marker2" and so on.

Return Value

Success or failure code.

Back to [Methods](#)

MarkerFunctionsMarkerRefValue

Description

Sets the value of the specified item to the value of the position of the marker.

Reference value is set to the response value of the marker position.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function MarkerFunctionsMarkerRefValue ( _  
    repeatedCapabilitiesID As String _  
) As Integer
```

C#

```
public int MarkerFunctionsMarkerRefValue(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int MarkerFunctionsMarkerRefValue(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function markerFunctionsMarkerRefValue(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index and Marker index for the active trace of channel i.e. "Channel1:Marker2" and so on. The physical names are supported along with the corresponding Channel index, Measurement index and Marker index for the n-th trace of channel i.e. "Channel1:Measurement3:Marker2" and so on.

Return Value

Success or failure code.

Back to [Methods](#)

MarkerFunctionsMarkerStart

Description

Sets the value of the specified item to the value of the position of the marker.

Sweep start value is set to the stimulus value of the marker position.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function MarkerFunctionsMarkerStart ( _  
    repeatedCapabilitiesID As String _  
) As Integer
```

C#

```
public int MarkerFunctionsMarkerStart(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int MarkerFunctionsMarkerStart(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function markerFunctionsMarkerStart(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index and Marker index for the active trace of channel i.e. "Channel1:Marker2" and so on. The physical names are supported along with the corresponding Channel index, Measurement index and Marker index for the n-th trace of channel i.e. "Channel1:Measurement3:Marker2" and so on.

Return Value

Success or failure code.

Back to [Methods](#)

MarkerFunctionsMarkerStop

Description

Sets the value of the specified item to the value of the position of the marker.

Sweep stop value is set to the stimulus value of the marker position.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function MarkerFunctionsMarkerStop ( _  
    repeatedCapabilitiesID As String _  
    ) As Integer
```

C#

```
public int MarkerFunctionsMarkerStop(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int MarkerFunctionsMarkerStop(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function markerFunctionsMarkerStop(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index and Marker index for the active trace of channel i.e. "Channel1:Marker2" and so on. The physical names are supported along with the corresponding Channel index, Measurement index and Marker index for the n-th trace of channel i.e. "Channel1:Measurement3:Marker2" and so on.

Return Value

Success or failure code.

Back to [Methods](#)

MarkerSetLimitLineResponseOffset

Description

Sets the value of the limit line offset along Y-axis to the active marker value.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function MarkerSetLimitLineResponseOffset ( _  
    repeatedCapabilitiesID As String _  
) As Integer
```

C#

```
public int MarkerSetLimitLineResponseOffset(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int MarkerSetLimitLineResponseOffset(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function markerSetLimitLineResponseOffset(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

Return Value

Success or failure code.

Back to [Methods](#)

MeasurementAutoRefValue

Description

Executes the auto reference function for the trace. The function automatically sets the reference level value.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function MeasurementAutoRefValue ( _  
    repeatedCapabilitiesID As String _  
) As Integer
```

C#

```
public int MeasurementAutoRefValue(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int MeasurementAutoRefValue(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function measurementAutoRefValue(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index and Measurement index i.e. "Channel1:Measurement3" and so on.

Return Value

Success or failure code.

Back to [Methods](#)

MeasurementAutoScale

Description

Executes the auto scale function for the trace.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function MeasurementAutoScale ( _  
    repeatedCapabilitiesID As String _  
) As Integer
```

C#

```
public int MeasurementAutoScale(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int MeasurementAutoScale(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function measurementAutoScale(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index and Measurement index i.e. "Channel1:Measurement3" and so on.

Return Value

Success or failure code.

Back to [Methods](#)

MeasurementDataToMemory

Description

Saves measurement trace data in memory.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function MeasurementDataToMemory ( _  
    repeatedCapabilitiesID As String _  
) As Integer
```

C#

```
public int MeasurementDataToMemory(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int MeasurementDataToMemory(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function measurementDataToMemory(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

Return Value

Success or failure code.

Back to [Methods](#)

MeasurementFetchComplex

Description

Returns the corrected data array. The corrected data array is the data, which processing is completed excluding the formatting as the last step. Such data represent S-parameter complex values.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function MeasurementFetchComplex ( _  
    repeatedCapabilitiesID As String, _  
    realValues As Double(), _  
    <OutAttribute> ByRef realreturnSize As Integer, _  
    imagValues As Double(), _  
    <OutAttribute> ByRef returnImagSize As Integer _  
    ) As Integer
```

C#

```
public int MeasurementFetchComplex(  
    string repeatedCapabilitiesID,  
    double[] realValues,  
    out int realreturnSize,  
    double[] imagValues,  
    out int returnImagSize  
)
```

Visual C++

```
public:  
    int MeasurementFetchComplex(  

```

Visual C++

```
String repeatedCapabilitiesID,  
double realValues[],  
int *realreturnSize,  
double imagValues[],  
int *returnImagSize  
)
```

JavaScript

```
function measurementFetchComplex(repeatedCapabilitiesID, realValues,  
realreturnSize, imagValues, returnImagSize);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

realValues

Type: **System.Double** []

The array of the real part of corrected measurement.

realreturnSize

Type: **System.Int32**

The actual number of elements filled by the query.

imagValues

Type: **System.Double** []

The array of the the imaginary part of corrected measurement.

[returnImagSize](#)

Type: **System.Int32**

The actual number of elements filled by the query.

Return Value

Success or failure code.

Back to [Methods](#)

MeasurementFetchFormatted

Description

Returns the formatted data array. The formatted data array is the data, which processing is completed including the formatting as the last step. Such data represent the data trace values as they are shown on the screen.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function MeasurementFetchFormatted ( _  
    repeatedCapabilitiesID As String, _  
    realValues As Double(), _  
    <OutAttribute> ByRef realreturnSize As Integer, _  
    imagValues As Double(), _  
    <OutAttribute> ByRef returnImagSize As Integer _  
    ) As Integer
```

C#

```
public int MeasurementFetchFormatted(  
    string repeatedCapabilitiesID,  
    double[] realValues,  
    out int realreturnSize,  
    double[] imagValues,  
    out int returnImagSize  
)
```

Visual C++

```
public:  
    int MeasurementFetchFormatted(  
        
```

Visual C++

```
String repeatedCapabilitiesID,  
double realValues[],  
int *realreturnSize,  
double imagValues[],  
int *returnImagSize  
)
```

JavaScript

```
function measurementFetchFormatted(repeatedCapabilitiesID, realValues,  
realreturnSize, imagValues, returnImagSize);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

realValues

Type: **System.Double** []

The array of the real number in rectangular format, real part in polar and Smith chart formats.

realreturnSize

Type: **System.Int32**

The actual number of elements filled by the query.

imagValues

Type: **System.Double** []

The array of 0 in rectangular format, imaginary part in polar and Smith chart formats.

returnImagSize

Type: **System.Int32**

The actual number of elements filled by the query.

Return Value

Success or failure code.

Back to [Methods](#)

MeasurementFetchMemoryComplex

Description

Returns the corrected memory array. The corrected memory array is the data, which processing is completed excluding the formatting as the last step. Such data represent S-parameter complex values.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function MeasurementFetchMemoryComplex ( _  
    repeatedCapabilitiesID As String, _  
    realValues As Double(), _  
    <OutAttribute> ByRef realreturnSize As Integer, _  
    imagValues As Double(), _  
    <OutAttribute> ByRef imagreturnSize As Integer _  
    ) As Integer
```

C#

```
public int MeasurementFetchMemoryComplex(  
    string repeatedCapabilitiesID,  
    double[] realValues,  
    out int realreturnSize,  
    double[] imagValues,  
    out int imagreturnSize  
)
```

Visual C++

```
public:  
int MeasurementFetchMemoryComplex(  

```

Visual C++

```
String repeatedCapabilitiesID,  
double realValues[],  
int *realreturnSize,  
double imagValues[],  
int *imagreturnSize  
)
```

JavaScript

```
function measurementFetchMemoryComplex(repeatedCapabilitiesID,  
realValues, realreturnSize, imagValues, imagreturnSize);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

realValues

Type: **System.Double** []

The array of the real part of corrected measurement memory.

realreturnSize

Type: **System.Int32**

The actual number of elements filled by the query.

imagValues

Type: **System.Double** []

The array of the imaginary part of corrected measurement memory.

imagreturnSize

Type: **System.Int32**

The actual number of elements filled by the query.

Return Value

Success or failure code.

Back to [Methods](#)

MeasurementFetchMemoryFormatted

Description

Returns the formatted memory array. The formatted memory array is the data, which processing is completed including the formatting as the last step. Such data represent the memory trace values as they are shown on the screen.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function MeasurementFetchMemoryFormatted ( _  
    repeatedCapabilitiesID As String, _  
    realValues As Double(), _  
    <OutAttribute> ByRef realreturnSize As Integer, _  
    imagValues As Double(), _  
    <OutAttribute> ByRef imagreturnSize As Integer _  
    ) As Integer
```

C#

```
public int MeasurementFetchMemoryFormatted(  
    string repeatedCapabilitiesID,  
    double[] realValues,  
    out int realreturnSize,  
    double[] imagValues,  
    out int imagreturnSize  
)
```

Visual C++

```
public:  
    int MeasurementFetchMemoryFormatted(  

```


Visual C++

```
String repeatedCapabilitiesID,  
double realValues[],  
int *realreturnSize,  
double imagValues[],  
int *imagreturnSize  
)
```

JavaScript

```
function measurementFetchMemoryFormatted(repeatedCapabilitiesID,  
realValues, realreturnSize, imagValues, imagreturnSize);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

realValues

Type: **System.Double** []

The array of the real number in rectangular format, real part in polar and Smith chart formats.

realreturnSize

Type: **System.Int32**

The actual number of elements filled by the query.

imagValues

Type: **System.Double** []

The array of 0 in rectangular format, imaginary part in polar and Smith chart formats.

imagreturnSize

Type: **System.Int32**

The actual number of elements filled by the query.

Return Value

Success or failure code.

Back to [Methods](#)

MeasurementFetchX

Description

Gets the X-axis values array. The X-axis values array is the frequency, power or time values array depending on the trace setup. The array contains real values. The array size is N, where N is the number of measurement points.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function MeasurementFetchX ( _  
    repeatedCapabilitiesID As String, _  
    retValues As Double(), _  
    <OutAttribute> ByRef returnSize As Integer _  
) As Integer
```

C#

```
public int MeasurementFetchX(  
    string repeatedCapabilitiesID,  
    double[] retValues,  
    out int returnSize  
)
```

Visual C++

```
public:  
int MeasurementFetchX(  
    String repeatedCapabilitiesID,  
    double retValues[],  
    int *returnSize  
)
```

JavaScript

```
function measurementFetchX(repeatedCapabilitiesID, retValues, returnSize);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

retValues

Type: **System.Double** []

The array of the the X-axis value.

returnSize

Type: **System.Int32**

The actual number of elements filled by the query.

Return Value

Success or failure code.

Back to [Methods](#)

MeasurementGetParameter

Description

Gets the measurement parameter of the trace.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function MeasurementGetParameter ( _  
    repeatedCapabilitiesID As String, _  
    <OutAttribute> ByRef measurementType As Integer, _  
    <OutAttribute> ByRef receiverPort As Integer, _  
    <OutAttribute> ByRef sourcePort As Integer _  
    ) As Integer
```

C#

```
public int MeasurementGetParameter(  
    string repeatedCapabilitiesID,  
    out int measurementType,  
    out int receiverPort,  
    out int sourcePort  
)
```

Visual C++

```
public:  
int MeasurementGetParameter(  
    String repeatedCapabilitiesID,  
    int *measurementType,  
    int *receiverPort,  
    int *sourcePort
```

Visual C++

```
)
```

JavaScript

```
function measurementGetParameter(repeatedCapabilitiesID,  
measurementType, receiverPort, sourcePort);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index and Measurement index i.e. "Channel1:Measurement3" and so on.

measurementType

Type: **System.Int32**

The type measurement: S – parameters, Test receivers, Reference receivers.

receiverPort

Type: **System.Int32**

The number of port from which the measurement is performed.

sourcePort

Type: **System.Int32**

The number of port which is the source.

Return Value

Success or failure code.

Back to [Methods](#)

MeasurementSetParameter

Description

Sets the measurement parameter of the trace.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function MeasurementSetParameter ( _  
    repeatedCapabilitiesID As String, _  
    measurementType As Integer, _  
    newreceiverPort As Integer, _  
    newsourcePort As Integer _  
    ) As Integer
```

C#

```
public int MeasurementSetParameter(  
    string repeatedCapabilitiesID,  
    int measurementType,  
    int newreceiverPort,  
    int newsourcePort  
)
```

Visual C++

```
public:  
int MeasurementSetParameter(  
    String repeatedCapabilitiesID,  
    int measurementType,  
    int newreceiverPort,  
    int newsourcePort
```

Visual C++

```
)
```

JavaScript

```
function measurementSetParameter(repeatedCapabilitiesID,  
measurementType, newreceiverPort, newsourcePort);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index and Measurement index i.e. "Channel1:Measurement3" and so on.

measurementType

Type: **System.Int32**

The type measurement: S – parameters, Test receivers, Reference receivers.

newreceiverPort

Type: **System.Int32**

The number of port from which the measurement is performed.

newsourcePort

Type: **System.Int32**

The number of port which is the source.

Return Value

Success or failure code.

Back to [Methods](#)

MeasurePortExtensionOpen

Description

Performs measurement of the standard "OPEN", automatically calculates and sets the parameters of the Port Extension.

When two consecutive measurements of "OPEN" is performed the results of these measurements are averaged.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function MeasurePortExtensionOpen ( _  
    repeatedCapabilitiesID As String _  
) As Integer
```

C#

```
public int MeasurePortExtensionOpen(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int MeasurePortExtensionOpen(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function measurePortExtensionOpen(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.

Return Value

Success or failure code.

Back to [Methods](#)

MeasurePortExtensionShort

Description

Performs measurement of the standard "SHORT" automatically calculates and sets the parameters of the Port Extension.

When two consecutive measurements of "SHORT" is performed the results of these measurements are averaged.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function MeasurePortExtensionShort ( _  
    repeatedCapabilitiesID As String _  
) As Integer
```

C#

```
public int MeasurePortExtensionShort(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int MeasurePortExtensionShort(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function measurePortExtensionShort(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.

Return Value

Success or failure code.

Back to [Methods](#)

NetworkAnalyzer(String, Boolean, Boolean)

Description

Opens the I/O session to the instrument.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Syntax

Visual Basic (Declaration)

```
Public Sub New ( _  
    resourceName As String, _  
    idQuery As Boolean, _  
    reset As Boolean _  
)
```

C#

```
public NetworkAnalyzer(  
    string resourceName,  
    bool idQuery,  
    bool reset  
)
```

Visual C++

```
public:  
NetworkAnalyzer(  
    String resourceName,  
    bool idQuery,  
    bool reset  
)
```

JavaScript

```
CMT.Instruments.NetworkAnalyzer = function(resourceName, idQuery, reset);
```

Parameters

resourceName

Type: **System.String**

VISA resource descriptor string

idQuery

Type: **System.Boolean**

Specifies whether to verify the ID of the instrument

reset

Type: **System.Boolean**

Specifies whether to reset the instrument

Exceptions

Exception	Condition
IOException	If the instrument fails to initialize, an <code>Mi.Driver.IOException("Error message")</code> is thrown.

Back to [Class NetworkAnalyzer](#)

NetworkAnalyzer(String, Boolean, Boolean, String)

Description

Opens the I/O session to the instrument with options.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Sub New ( _  
    resourceName As String, _  
    idQuery As Boolean, _  
    reset As Boolean, _  
    optionsString As String _  
)
```

C#

```
public NetworkAnalyzer(  
    string resourceName,  
    bool idQuery,  
    bool reset,  
    string optionsString  
)
```

Visual C++

```
public:  
NetworkAnalyzer(  
    String resourceName,  
    bool idQuery,  
    bool reset,  
    String optionsString
```

Visual C++

```
)
```

JavaScript

```
CMT.Instruments.NetworkAnalyzer = function(resourceName, idQuery, reset, optionsString);
```

Parameters

resourceName

Type: **System.String**

VISA resource descriptor string

idQuery

Type: **System.Boolean**

Specifies whether to verify the ID of the instrument

reset

Type: **System.Boolean**

Specifies whether to reset the instrument

optionsString

Type: **System.String**

The user can use the OptionsString parameter to specify the initial values of certain MI inherent attributes for the session. For a detail description of OptionsString parameter see in [Instantiating the MI.NET Driver](#).

Exceptions

Exception	Condition
IOException	If the instrument fails to initialize, an <code>Mi.Driver.IOException("Error message")</code> is thrown.

Back to [Class NetworkAnalyzer](#)

PerformCalibrationAction

Description

Performs one calibration action on the specified ports or the type of calibration action. It is recommended to use WaitForOperationComplete function to check if operation is completed.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function PerformCalibrationAction ( _  
    repeatedCapabilitiesID As String, _  
    receiverPort As Integer, _  
    sourcePort As Integer, _  
    actionType As Integer _  
) As Integer
```

C#

```
public int PerformCalibrationAction(  
    string repeatedCapabilitiesID,  
    int receiverPort,  
    int sourcePort,  
    int actionType  
)
```

Visual C++

```
public:  
int PerformCalibrationAction(  
    String repeatedCapabilitiesID,  
    int receiverPort,
```

Visual C++

```
int sourcePort,  
int actionType  
)
```

JavaScript

```
function performCalibrationAction(repeatedCapabilitiesID, receiverPort,  
sourcePort, actionType);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

receiverPort

Type: **System.Int32**

The number of port from which the measurement is performed.

sourcePort

Type: **System.Int32**

The number of port which is the source.

actionType

Type: **System.Int32**

Enum type of action:

[CalibrationActionOpen](#)

[CalibrationActionShort](#)

[CalibrationActionLoad](#)

[CalibrationActionThru](#)

[CalibrationActionIsolation](#)

[CalibrationActionPowerCal](#)

[CalibrationActionTestReceiverCal](#)

[CalibrationActionReferenceReceiverCal](#)

Return Value

Success or failure code.

Back to [Methods](#)

PerformUnkThru2PortCal

Description

Completes the full 2-port calibration between the specified ports provided so that each port was calibrated using full 1-port calibration: • Measures an arbitrary thru between the ports;

- Calculates the error terms Et and EI using the unknown thru algorithm
- Saves the Et and EI error terms to the existing calibration getting the full 2-port calibration from the two 1-port calibrations. If the full 2-port calibration already existed between the specified ports, updates the Et and EI error terms. It is recommended to use CmtNA_WaitForOperationComplete function to check if operation is completed

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function PerformUnkThru2PortCal ( _  
    repeatedCapabilitiesID As String, _  
    port1 As Integer, _  
    port2 As Integer _  
) As Integer
```

C#

```
public int PerformUnkThru2PortCal(  
    string repeatedCapabilitiesID,  
    int port1,  
    int port2  
)
```

Visual C++

```
public:  
    int PerformUnkThru2PortCal(  

```

Visual C++

```
String repeatedCapabilitiesID,  
int port1,  
int port2  
)
```

JavaScript

```
function performUnkThru2PortCal(repeatedCapabilitiesID, port1, port2);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

port1

Type: **System.Int32**

The first port number from 1 to 4.

port2

Type: **System.Int32**

The second port number from 1 to 4.

Return Value

Success or failure code.

Back to [Methods](#)

PowerSensorZeroing

Description

Executes zeroing procedure of the power sensor.

Although the Analyzer automatically turns off the RF power during this procedure, it is recommended to disconnect the power sensor from the analyzer port.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function PowerSensorZeroing ( _  
    repeatedCapabilitiesID As String _  
) As Integer
```

C#

```
public int PowerSensorZeroing(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int PowerSensorZeroing(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function powerSensorZeroing(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Return Value

Success or failure code.

Back to [Methods](#)

PrintOut

Description

Prints out the image displayed on the screen without previewing.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function PrintOut As Integer
```

C#

```
public int PrintOut()
```

Visual C++

```
public:  
int PrintOut()
```

JavaScript

```
function printOut();
```

Return Value

Success or failure code.

Back to [Methods](#)

QueryMarkerMathBandwidth

Description

Gets the bandwidth search result.

The bandwidth search can performed relatively to the marker, or relatively to the absolute maximum value of the trace (in this case the number of the marker is ignored), that is set by [MarkerMathBandwidthSearchRef](#)

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function QueryMarkerMath bandwidth ( _  
    repeatedCapabilitiesID As String, _  
    <OutAttribute> ByRef bandwidth As Double, _  
    <OutAttribute> ByRef centerFreq As Double, _  
    <OutAttribute> ByRef qvalue As Double, _  
    <OutAttribute> ByRef loss As Double _  
    ) As Integer
```

C#

```
public int QueryMarkerMath bandwidth(  
    string repeatedCapabilitiesID,  
    out double bandwidth,  
    out double centerFreq,  
    out double qvalue,  
    out double loss  
)
```

Visual C++

```
public:
```

Visual C++

```
int QueryMarkerMath bandwidth(  
    String repeatedCapabilitiesID,  
    double *bandwidth,  
    double *centerFreq,  
    double *qvalue,  
    double *loss  
)
```

JavaScript

```
function queryMarkerMathbandwidth(repeatedCapabilitiesID, bandwidth,  
    centerFreq, qvalue, loss);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index and Marker index for the active trace of channel i.e. "Channel1:Marker2" and so on. The physical names are supported along with the corresponding Channel index and Measurement index and Marker index for the n–th trace of channel i.e. "Channel1:Measurement3:Marker2" and so on.

bandwidth

Type: **System.Double**

The bandwidth value.

centerFreq

Type: **System.Double**

The center frequency value.

qvalue

Type: **System.Double**

The Q value.

loss

Type: **System.Double**

The loss value.

Return Value

Success or failure code.

Back to [Methods](#)

QueryMarkerMathFlatness

Description

Gets FLATNESS function data array. The FLATNESS function is applied within the range determined by two markers.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function QueryMarkerMathFlatness ( _  
    repeatedCapabilitiesID As String, _  
    <OutAttribute> ByRef span As Double, _  
    <OutAttribute> ByRef gain As Double, _  
    <OutAttribute> ByRef slope As Double, _  
    <OutAttribute> ByRef flatness As Double _  
    ) As Integer
```

C#

```
public int QueryMarkerMathFlatness(  
    string repeatedCapabilitiesID,  
    out double span,  
    out double gain,  
    out double slope,  
    out double flatness  
)
```

Visual C++

```
public:  
int QueryMarkerMathFlatness(  
    String repeatedCapabilitiesID,
```

Visual C++

```
double *span,  
double *gain,  
double *slope,  
double *flatness  
)
```

JavaScript

```
function queryMarkerMathFlatness(repeatedCapabilitiesID, span, gain, slope,  
flatness);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

span

Type: **System.Double**

The span value.

gain

Type: **System.Double**

The gain value.

slope

Type: **System.Double**

The slope value.

flatness

Type: **System.Double**

The flatness value.

Return Value

Success or failure code.

Back to [Methods](#)

QueryMarkerMathStatistics

Description

Gets the math statistics values.

The statistics function is applied either over the whole range, or within the range specified by [MarkerMathStatistics](#) (the range limits are determined by two markers).

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function QueryMarkerMathStatistics ( _  
    repeatedCapabilitiesID As String, _  
    <OutAttribute> ByRef mean As Double, _  
    <OutAttribute> ByRef sdev As Double, _  
    <OutAttribute> ByRef peakPeak As Double _  
) As Integer
```

C#

```
public int QueryMarkerMathStatistics(  
    string repeatedCapabilitiesID,  
    out double mean,  
    out double sdev,  
    out double peakPeak  
)
```

Visual C++

```
public:  
int QueryMarkerMathStatistics(  
    String repeatedCapabilitiesID,  
    double *mean,
```


Visual C++

```
double *sdev,  
double *peakPeak  
)
```

JavaScript

```
function queryMarkerMathStatistics(repeatedCapabilitiesID, mean, sdev,  
peakPeak);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

mean

Type: **System.Double**

The mean value.

sdev

Type: **System.Double**

The standard deviation value.

peakPeak

Type: **System.Double**

The peak-to-peak value (difference between the maximum value and the minimum value).

Return Value

Success or failure code.

Back to [Methods](#)

QueryMarkerTableData

Description

Gets the data array of all turned ON markers.

The array size is $3N + 1$, where N is the number of turned ON markers including the reference marker.

If the reference marker is turned ON the last three elements of the array contain the reference marker data and the rest elements of array contain the relative values.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function QueryMarkerTableData ( _  
    repeatedCapabilitiesID As String, _  
    <OutAttribute> ByRef markerONCount As Integer, _  
    stimulusValues As Double(), _  
    realValues As Double(), _  
    imageValues As Double() _  
    ) As Integer
```

C#

```
public int QueryMarkerTableData(  
    string repeatedCapabilitiesID,  
    out int markerONCount,  
    double[] stimulusValues,  
    double[] realValues,  
    double[] imageValues  
)
```

Visual C++

```
public:
int QueryMarkerTableData(
    String repeatedCapabilitiesID,
    int *markerONCount,
    double stimulusValues[],
    double realValues[],
    double imageValues[]
)
```

JavaScript

```
function queryMarkerTableData(repeatedCapabilitiesID, markerONCount,
stimulusValues, realValues, imageValues);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

markerONCount

Type: **System.Int32**

The number of turned ON markers including the reference marker (N).

stimulusValues

Type: **System.Double** []

The stimulus value of the n–th marker.

realValues

Type: **System.Double** []

The real data in rectangular format, real part in polar and Smith chart formats of the n–th marker.

imageValues

Type: **System.Double** []

0 in rectangular format, imaginary part in polar and Smith chart formats of the n–th marker.

Return Value

Success or failure code.

Back to [Methods](#)

RecallChannelFromRegister

Description

Recalls the Analyzer state for the active channel.

The file must be saved in one of the four memory registers by [SaveChannelToRegister](#) function.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function RecallChannelFromRegister ( _  
    repeatedCapabilitiesID As String, _  
    stateRegister As Integer _  
    ) As Integer
```

C#

```
public int RecallChannelFromRegister(  
    String repeatedCapabilitiesID,  
    int stateRegister  
)
```

Visual C++

```
public:  
int RecallChannelFromRegister(  
    String repeatedCapabilitiesID,  
    int stateRegister  
)
```

JavaScript

```
function recallChannelFromRegister(repeatedCapabilitiesID, stateRegister);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

stateRegister

Type: **System.Int32**

The memory register whence recall the Analyzer state for the active channel:

[ChannelstateRegisterA](#)

[ChannelstateRegisterB](#)

[ChannelstateRegisterC](#)

[ChannelstateRegisterD](#)

Return Value

Success or failure code.

Back to [Methods](#)

RecallFromFile

Description

Recalls the specified Analyzer state file.

The file must be saved by [SaveStateToFile](#) function.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function RecallFromFile ( _  
    fileName As String _  
) As Integer
```

C#

```
public int RecallFromFile(  
    string fileName  
)
```

Visual C++

```
public:  
int RecallFromFile(  
    String fileName  
)
```

JavaScript

```
function recallFromFile(fileName);
```

Parameters

fileName

Type: **System.String**

The file path into which to save the data. If the full path of the file is not specified, the \State subdirectory of the application directory will be searched for the file. The Analyzer state file has *.sta extension by default.

Return Value

Success or failure code.

Back to [Methods](#)

RecallSegmentTable

Description

Recalls the segment table file.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function RecallSegmentTable ( _  
    repeatedCapabilitiesID As String, _  
    fileName As String _  
) As Integer
```

C#

```
public int RecallSegmentTable(  
    string repeatedCapabilitiesID,  
    string fileName  
)
```

Visual C++

```
public:  
int RecallSegmentTable(  
    String repeatedCapabilitiesID,  
    String fileName  
)
```

JavaScript

```
function recallSegmentTable(repeatedCapabilitiesID, fileName);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

Pass VI_NULL or empty string if operation does not apply to a repeated capability.

fileName

Type: **System.String**

The file path into which to save the data.

Return Value

Success or failure code.

Back to [Methods](#)

Reset

Description

Resets the device to a known initialization state.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function Reset As Integer
```

C#

```
public int Reset()
```

Visual C++

```
public:  
int Reset()
```

JavaScript

```
function reset();
```

Return Value

Success or failure code.

Back to [Methods](#)

ResetCalibration

Description

Clears the calibration coefficient table.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function ResetCalibration ( _  
    repeatedCapabilitiesID As String _  
) As Integer
```

C#

```
public int ResetCalibration(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int ResetCalibration(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function resetCalibration(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Return Value

Success or failure code.

Back to [Methods](#)

ResetWithDefaults

Description

Resets the device to a known initialization state with defaults as specified for the device in the optionString parameter.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function ResetWithDefaults As Integer
```

C#

```
public int ResetWithDefaults()
```

Visual C++

```
public:  
int ResetWithDefaults()
```

JavaScript

```
function resetWithDefaults();
```

Return Value

Success or failure code.

Back to [Methods](#)

RestoreLimitTable

Description

Recalls the limit table file. The file must be saved by [SaveLimitTable](#) function.

The function is used for the active trace of the active channel.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function RestoreLimitTable ( _  
    repeatedCapabilitiesID As String, _  
    fileName As String _  
) As Integer
```

C#

```
public int RestoreLimitTable(  
    string repeatedCapabilitiesID,  
    string fileName  
)
```

Visual C++

```
public:  
int RestoreLimitTable(  
    String repeatedCapabilitiesID,  
    String fileName  
)
```

JavaScript

```
function restoreLimitTable(repeatedCapabilitiesID, fileName);
```


Parameters

repeatedCapabilitiesID

Type: **System.String**

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

fileName

Type: **System.String**

The file path into which to save the data. If the full path of the file is not specified, the \Limit subdirectory of the application directory will be searched for the file. The file has *.lim extension by default.

Return Value

Success or failure code.

Back to [Methods](#)

RestoreRippleLimitTable

Description

Recalls the ripple limit table file.

The file must be saved by [SaveRippleLimitTable](#) function.

The function is used for the active trace of the active channel.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function RestoreRippleLimitTable ( _  
    repeatedCapabilitiesID As String, _  
    fileName As String _  
    ) As Integer
```

C#

```
public int RestoreRippleLimitTable(  
    String repeatedCapabilitiesID,  
    String fileName  
)
```

Visual C++

```
public:  
int RestoreRippleLimitTable(  
    String repeatedCapabilitiesID,  
    String fileName  
)
```

JavaScript

```
function restoreRippleLimitTable(repeatedCapabilitiesID, fileName);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

fileName

Type: **System.String**

The file path into which to save the data. If the full path of the file is not specified, the \Limit subdirectory of the application directory will be searched for the file. The file has *.lim extension by default.

Return Value

Success or failure code.

Back to [Methods](#)

RevisionQuery

Description

Queries the revision of the device.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function RevisionQuery ( _  
    <OutAttribute> ByRef driverRevision As String, _  
    <OutAttribute> ByRef firmwareRevision As String _  
) As Integer
```

C#

```
public int RevisionQuery(  
    out String driverRevision,  
    out String firmwareRevision  
)
```

Visual C++

```
public:  
int RevisionQuery(  
    String driverRevision[],  
    String firmwareRevision[]  
)
```

JavaScript

```
function revisionQuery(driverRevision, firmwareRevision);
```

Parameters

driverRevision

Type: **System.String**

Returns the revision of the IVI specific driver.

firmwareRevision

Type: **System.String**

Returns the firmware revision of the instrument

Return Value

Success or failure code.

Back to [Methods](#)

SaveChannelToRegister

Description

Saves the Analyzer state of the items set for the active channel into one of the four memory registers.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SaveChannelToRegister ( _  
    repeatedCapabilitiesID As String, _  
    stateRegister As Integer _  
    ) As Integer
```

C#

```
public int SaveChannelToRegister(  
    string repeatedCapabilitiesID,  
    int stateRegister  
)
```

Visual C++

```
public:  
int SaveChannelToRegister(  
    String repeatedCapabilitiesID,  
    int stateRegister  
)
```

JavaScript

```
function saveChannelToRegister(repeatedCapabilitiesID, stateRegister);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

stateRegister

Type: **System.Int32**

The memory register for saving the file:

[ChannelstateRegisterA](#)

[ChannelstateRegisterB](#)

[ChannelstateRegisterC](#)

[ChannelstateRegisterD](#)

Return Value

Success or failure code.

Back to [Methods](#)

SaveImage

Description

Saves the display image in BMP or PNG format into a file.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SaveImage ( _  
    fileName As String _  
) As Integer
```

C#

```
public int SaveImage(  
    string fileName  
)
```

Visual C++

```
public:  
intSaveImage(  
    String fileName  
)
```

JavaScript

```
function saveImage(fileName);
```

Parameters

fileName

Type: **System.String**

The file path into which to save the data. If the full path of the file is not specified, the \Image subdirectory of the application directory will be searched for the file. If the file has *.png extension, the file had PNG format, in all the other cases the file has BMP format.

Return Value

Success or failure code.

Back to [Methods](#)

SaveLimitTable

Description

Recalls the limit table file.

The function is used for the active trace of the active channel.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SaveLimitTable ( _  
    repeatedCapabilitiesID As String, _  
    fileName As String _  
    ) As Integer
```

C#

```
public int SaveLimitTable(  
    string repeatedCapabilitiesID,  
    string fileName  
)
```

Visual C++

```
public:  
int SaveLimitTable(  
    String repeatedCapabilitiesID,  
    String fileName  
)
```

JavaScript

```
function saveLimitTable(repeatedCapabilitiesID, fileName);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

fileName

Type: **System.String**

The file path into which to save the data. If the full path of the file is not specified, the \Limit subdirectory of the application directory will be searched for the file. The file has *.lim extension by default.

Return Value

Success or failure code.

Back to [Methods](#)

SaveRippleLimitTable

Description

Saves the ripple limit table into a file.

The function is used for the active trace of the active channel.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SaveRippleLimitTable ( _  
    repeatedCapabilitiesID As String, _  
    fileName As String _  
    ) As Integer
```

C#

```
public int SaveRippleLimitTable(  
    string repeatedCapabilitiesID,  
    string fileName  
)
```

Visual C++

```
public:  
int SaveRippleLimitTable(  
    String repeatedCapabilitiesID,  
    String fileName  
)
```

JavaScript

```
function saveRippleLimitTable(repeatedCapabilitiesID, fileName);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

fileName

Type: **System.String**

The file path into which to save the data. If the full path of the file is not specified, the \Limit subdirectory of the application directory will be searched for the file. The file has *.lim extension by default.

Return Value

Success or failure code.

Back to [Methods](#)

SaveSegmentTable

Description

Saves the segment table in a file.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SaveSegmentTable ( _  
    repeatedCapabilitiesID As String, _  
    fileName As String _  
) As Integer
```

C#

```
public int SaveSegmentTable(  
    string repeatedCapabilitiesID,  
    string fileName  
)
```

Visual C++

```
public:  
int SaveSegmentTable(  
    String repeatedCapabilitiesID,  
    String fileName  
)
```

JavaScript

```
function saveSegmentTable(repeatedCapabilitiesID, fileName);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

Pass VI_NULL or empty string if operation does not apply to a repeated capability.

fileName

Type: **System.String**

The file path into which to save the data.

Return Value

Success or failure code.

Back to [Methods](#)

SaveStateToFile

Description

Saves the Analyzer state into a file.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SaveStateToFile ( _  
    fileName As String _  
) As Integer
```

C#

```
public int SaveStateToFile(  
    string fileName  
)
```

Visual C++

```
public:  
int SaveStateToFile(  
    String fileName  
)
```

JavaScript

```
function saveStateToFile(fileName);
```

Parameters

fileName

Type: **System.String**

The file path into which to save the data. If the full path of the file is not specified, the \State subdirectory of the application directory will be searched for the file. The state file has *.sta extension by default.

Return Value

Success or failure code.

Back to [Methods](#)

SaveTouchstoneFile

Description

Saves the measured S-parameters of the active channel into a Touchstone file.

The file type (1 port to 4 port) is set by [SetTouchstoneFileType\(String, Int32, Int32\[\]\)](#) function.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SaveTouchstoneFile ( _  
    repeatedCapabilitiesID As String, _  
    fileName As String _  
    ) As Integer
```

C#

```
public int SaveTouchstoneFile(  
    string repeatedCapabilitiesID,  
    string fileName  
)
```

Visual C++

```
public:  
int SaveTouchstoneFile(  
    String repeatedCapabilitiesID,  
    String fileName  
)
```

JavaScript

```
function saveTouchstoneFile(repeatedCapabilitiesID, fileName);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

fileName

Type: **System.String**

The file path into which to save the data. If the full path of the file is not specified, the \FixtureSim subdirectory of the application directory will be searched for the file. The file has *.snp extension by default.

Return Value

Success or failure code.

Back to [Methods](#)

SaveTraceData

Description

Saves the CSV formatted data into a file.

The function is used for the active trace of the active channel.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SaveTraceData ( _  
    repeatedCapabilitiesID As String, _  
    fileName As String _  
    ) As Integer
```

C#

```
public int SaveTraceData(  
    string repeatedCapabilitiesID,  
    string fileName  
)
```

Visual C++

```
public:  
int SaveTraceData(  
    String repeatedCapabilitiesID,  
    String fileName  
)
```

JavaScript

```
function saveTraceData(repeatedCapabilitiesID, fileName);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

fileName

Type: **System.String**

The file path into which to save the data. If the full path of the file is not specified, the \CSV subdirectory of the application directory will be searched for the file. The file has *.csv extension by default.

Return Value

Success or failure code.

Back to [Methods](#)

SelfTest

Description

Performs an instrument self test, waits for the instrument to complete the test, and queries the instrument for the results.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SelfTest ( _  
    <OutAttribute> ByRef testResult As Integer, _  
    <OutAttribute> ByRef testMessage As String _  
    ) As Integer
```

C#

```
public int SelfTest(  
    out int testResult,  
    out string testMessage  
)
```

Visual C++

```
public:  
int SelfTest(  
    int *testResult,  
    String *testMessage  
)
```

JavaScript

```
function selfTest(testResult, testMessage);
```

Parameters

testResult

Type: **System.Int32**

The numeric result from the self test operation. 0 = no error (test passed).

testMessage

Type: **System.String**

The self test status message.

Return Value

Success or failure code.

Back to [Methods](#)

SetAttributeViBoolean

Description

Sets the value of a ViBoolean attribute.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SetAttributeViBoolean ( _  
    repeatedCapabilitiesID As String, _  
    attributeID As Integer, _  
    value As Boolean _  
) As Integer
```

C#

```
public int SetAttributeViBoolean(  
    string repeatedCapabilitiesID,  
    int attributeID,  
    bool value  
)
```

Visual C++

```
public:  
int SetAttributeViBoolean(  
    String repeatedCapabilitiesID,  
    int attributeID,  
    bool value  
)
```


JavaScript

```
function setAttributeViBoolean(repeatedCapabilitiesID, attributeID, value);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The repeated capability identifier

attributeID

Type: **System.Int32**

Pass the ID of an attribute. See static class Properties for attribute values.

value

Type: **System.Boolean**

Pass the value to which you want to set the attribute.

Return Value

Success or failure code.

Back to [Methods](#)

SetAttributeViInt32

Description

Sets the value of a ViInt32 attribute.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SetAttributeViInt32 ( _  
    repeatedCapabilitiesID As String, _  
    attributeID As Integer, _  
    value As Integer _  
) As Integer
```

C#

```
public int SetAttributeViInt32(  
    string repeatedCapabilitiesID,  
    int attributeID,  
    int value  
)
```

Visual C++

```
public:  
int SetAttributeViInt32(  
    String repeatedCapabilitiesID,  
    int attributeID,  
    int value  
)
```

JavaScript

```
function setAttributeVInt32(repeatedCapabilitiesID, attributeID, value);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The repeated capability identifier

attributeID

Type: **System.Int32**

Pass the ID of an attribute. See static class Properties for attribute values.

value

Type: **System.Int32**

Pass the value to which you want to set the attribute.

Return Value

Success or failure code.

Back to [Methods](#)

SetAttributeViReal64

Description

Sets the value of a ViReal64 attribute.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SetAttributeViReal64 ( _  
    repeatedCapabilitiesID As String, _  
    attributeID As Integer, _  
    value As Double _  
) As Integer
```

C#

```
public int SetAttributeViReal64(  
    string repeatedCapabilitiesID,  
    int attributeID,  
    double value  
)
```

Visual C++

```
public:  
int SetAttributeViReal64(  
    String repeatedCapabilitiesID,  
    int attributeID,  
    double value  
)
```

JavaScript

```
function setAttributeViReal64(repeatedCapabilitiesID, attributeID, value);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The repeated capability identifier

attributeID

Type: **System.Int32**

Pass the ID of an attribute. See static class Attributes for values.

value

Type: **System.Double**

Pass the value to which you want to set the attribute.

Return Value

Success or failure code.

Back to [Methods](#)

SetAttributeViString

Description

Sets the value of a ViString attribute.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SetAttributeViString ( _  
    repeatedCapabilitiesID As String, _  
    attributeID As Integer, _  
    value As String _  
) As Integer
```

C#

```
public int SetAttributeViString(  
    string repeatedCapabilitiesID,  
    int attributeID,  
    string value  
)
```

Visual C++

```
public:  
int SetAttributeViString(  
    String repeatedCapabilitiesID,  
    int attributeID,  
    String value  
)
```

JavaScript

```
function setAttributeViString(repeatedCapabilitiesID, attributeID, value);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The repeated capability identifier

attributeID

Type: **System.Int32**

Pass the ID of an attribute. See static class Properties for attribute values.

value

Type: **System.String**

Pass the value to which you want to set the attribute.

Return Value

Success or failure code.

Back to [Methods](#)

SetCalKitOrderLoad

Description

Sets the number of the calibration standard of the load type, used for the measurement of the specified port.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SetCalKitOrderLoad ( _  
    port As Integer, _  
    stdNumber As Integer _  
) As Integer
```

C#

```
public int SetCalKitOrderLoad(  
    int port,  
    int stdNumber  
)
```

Visual C++

```
public:  
int SetCalKitOrderLoad(  
    int port,  
    int stdNumber  
)
```

JavaScript

```
function setCalKitOrderLoad(port, stdNumber);
```

Parameters

port

Type: **System.Int32**

The number of port from which the measurement is performed.

stdNumber

Type: **System.Int32**

The number of the calibration standard.

Return Value

Success or failure code.

Back to [Methods](#)

SetCalKitOrderOpen

Description

Sets the number of the calibration standard of the open type, used for the measurement of the specified port.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SetCalKitOrderOpen ( _  
    port As Integer, _  
    stdNumber As Integer _  
) As Integer
```

C#

```
public int SetCalKitOrderOpen(  
    int port,  
    int stdNumber  
)
```

Visual C++

```
public:  
int SetCalKitOrderOpen(  
    int port,  
    int stdNumber  
)
```

JavaScript

```
function setCalKitOrderOpen(port, stdNumber);
```

Parameters

port

Type: **System.Int32**

The number of port from which the measurement is performed.

stdNumber

Type: **System.Int32**

The number of the calibration standard.

Return Value

Success or failure code.

Back to [Methods](#)

SetCalKitOrderShort

Description

Sets the number of the calibration standard of the short type, used for the measurement of the specified port.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SetCalKitOrderShort ( _  
    port As Integer, _  
    stdNumber As Integer _  
) As Integer
```

C#

```
public int SetCalKitOrderShort(  
    int port,  
    int stdNumber  
)
```

Visual C++

```
public:  
int SetCalKitOrderShort(  
    int port,  
    int stdNumber  
)
```

JavaScript

```
function setCalKitOrderShort(port, stdNumber);
```

Parameters

port

Type: **System.Int32**

The number of port from which the measurement is performed.

stdNumber

Type: **System.Int32**

The number of the calibration standard.

Return Value

Success or failure code.

Back to [Methods](#)

SetCalKitOrderThru

Description

Sets the number of the calibration standard of the thru type, used for the measurement between the source and receiver ports.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SetCalKitOrderThru ( _  
    receiverPort As Integer, _  
    sourcePort As Integer, _  
    stdNumber As Integer _  
) As Integer
```

C#

```
public int SetCalKitOrderThru(  
    int receiverPort,  
    int sourcePort,  
    int stdNumber  
)
```

Visual C++

```
public:  
int SetCalKitOrderThru(  
    int receiverPort,  
    int sourcePort,  
    int stdNumber  
)
```

JavaScript

```
function setCalKitOrderThru(receiverPort, sourcePort, stdNumber);
```

Parameters

receiverPort

Type: **System.Int32**

The number of port from which the measurement is performed.

sourcePort

Type: **System.Int32**

The number of port which is the source.

stdNumber

Type: **System.Int32**

The number of the calibration standard.

Return Value

Success or failure code.

Back to [Methods](#)

SetCalKitOrderTrlLine

Description

Sets the number of the calibration standard of the TRL line type, used for the measurement between the source and receiver ports.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SetCalKitOrderTrlLine ( _  
    receiverPort As Integer, _  
    sourcePort As Integer, _  
    stdNumber As Integer _  
) As Integer
```

C#

```
public int SetCalKitOrderTrlLine(  
    int receiverPort,  
    int sourcePort,  
    int stdNumber  
)
```

Visual C++

```
public:  
int SetCalKitOrderTrlLine(  
    int receiverPort,  
    int sourcePort,  
    int stdNumber  
)
```


JavaScript

```
function setCalKitOrderTrlLine(receiverPort, sourcePort, stdNumber);
```

Parameters

receiverPort

Type: **System.Int32**

The number of port from which the measurement is performed.

sourcePort

Type: **System.Int32**

The number of port which is the source.

stdNumber

Type: **System.Int32**

The number of the calibration standard.

Return Value

Success or failure code.

Back to [Methods](#)

SetCalKitOrderTrlReflect

Description

Sets the number of the calibration standard of the TRL Reflect type, used for the measurement of the specified port.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SetCalKitOrderTrlReflect ( _  
    port As Integer, _  
    stdNumber As Integer _  
) As Integer
```

C#

```
public int SetCalKitOrderTrlReflect(  
    int port,  
    int stdNumber  
)
```

Visual C++

```
public:  
int SetCalKitOrderTrlReflect(  
    int port,  
    int stdNumber  
)
```

JavaScript

```
function setCalKitOrderTrlReflect(port, stdNumber);
```

Parameters

port

Type: **System.Int32**

The number of port from which the measurement is performed.

stdNumber

Type: **System.Int32**

The number of the calibration standard.

Return Value

Success or failure code.

Back to [Methods](#)

SetCalKitOrderTrlThru

Description

Sets the number of the calibration standard of the thru type, used for the measurement between the source and receiver ports.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SetCalKitOrderThru ( _  
    receiverPort As Integer, _  
    sourcePort As Integer, _  
    stdNumber As Integer _  
) As Integer
```

C#

```
public int SetCalKitOrderThru(  
    int receiverPort,  
    int sourcePort,  
    int stdNumber  
)
```

Visual C++

```
public:  
int SetCalKitOrderThru(  
    int receiverPort,  
    int sourcePort,  
    int stdNumber  
)
```

JavaScript

```
function setCalKitOrderThru(receiverPort, sourcePort, stdNumber);
```

Parameters

receiverPort

Type: **System.Int32**

The number of port from which the measurement is performed.

sourcePort

Type: **System.Int32**

The number of port which is the source.

stdNumber

Type: **System.Int32**

The number of the calibration standard.

Return Value

Success or failure code.

Back to [Methods](#)

SetCalStandardS1PData

Description

Sets the data array of the data-based calibration standard. The first element of the array is 1 and determines the number of ports of the calibration standard.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SetCalStandardS1PData ( _  
    repeatedCapabilitiesID As String, _  
    stimulusValues As Double(), _  
    realS11Values As Double(), _  
    imageS11Values As Double() _  
    ) As Integer
```

C#

```
public int SetCalStandardS1PData(  
    string repeatedCapabilitiesID,  
    double[] stimulusValues,  
    double[] realS11Values,  
    double[] imageS11Values  
)
```

Visual C++

```
public:  
int SetCalStandardS1PData(  
    String repeatedCapabilitiesID,  
    double stimulusValues[],  
    double realS11Values[],
```

Visual C++

```
double imageS11Values[]  
)
```

JavaScript

```
function setCalStandardS1PData(repeatedCapabilitiesID, stimulusValues,  
realS11Values, imageS11Values);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.

stimulusValues

Type: **System.Double** []

The array of stimulus value.

realS11Values

Type: **System.Double** []

The array of value real part S11.

imageS11Values

Type: **System.Double** []

The array of value image part S11.

Return Value

Success or failure code.

Back to [Methods](#)

SetCalStandardS2PData

Description

Sets the data array of the data-based calibration standard. The first element of the array is 2 and determines the number of ports of the calibration standard.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SetCalStandardS2PData ( _  
    repeatedCapabilitiesID As String, _  
    stimulusValues As Double(), _  
    realS11Values As Double(), _  
    imageS11Values As Double(), _  
    realS21Values As Double(), _  
    imageS21Values As Double(), _  
    realS12Values As Double(), _  
    imageS12Values As Double(), _  
    realS22Values As Double(), _  
    imageS22Values As Double() _  
    ) As Integer
```

C#

```
public int SetCalStandardS2PData(  
    string repeatedCapabilitiesID,  
    double[] stimulusValues,  
    double[] realS11Values,  
    double[] imageS11Values,  
    double[] realS21Values,  
    double[] imageS21Values,
```


C#

```
double[] realS12Values,  
double[] imageS12Values,  
double[] realS22Values,  
double[] imageS22Values  
)
```

Visual C++

```
public:  
intSetCalStandardS2PData(  
    String repeatedCapabilitiesID,  
    double stimulusValues[]  
    double realS11Values[],  
    double imageS11Values[],  
    double realS21Values[],  
    double imageS21Values[],  
    double realS12Values[],  
    double imageS12Values[],  
    double realS22Values[],  
    double imageS22Values[]  
)
```

JavaScript

```
function setCalStandardS2PData(repeatedCapabilitiesID, stimulusValues,  
realS11Values, imageS11Values, realS21Values, imageS21Values,  
realS12Values, imageS12Values, realS22Values, imageS22Values);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.

stimulusValues

Type: **System.Double** []

The array of stimulus value.

realS11Values

Type: **System.Double** []

The array of value real part S11.

imageS11Values

Type: **System.Double** []

The array of value image part S11.

realS21Values

Type: **System.Double** []

The array of value real part S21.

imageS21Values

Type: **System.Double** []

The array of value image part S21.

realS12Values

Type: **System.Double** []

The array of value real part S12.

imageS12Values

Type: **System.Double** []

The array of value image part S12.

realS22Values

Type: **System.Double** []

The array of value real part S22.

imageS22Values

Type: **System.Double** []

The array of value image part S22.

Return Value

Success or failure code.

Back to [Methods](#)

SetLimitTestData

Description

Sets the data array, which is the limit line in the limit test function.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SetLimitTestData ( _  
    repeatedCapabilitiesID As String, _  
    limitLineType As Integer(), _  
    beginLimitArgument As Double(), _  
    endLimitArgument As Double(), _  
    beginlimitValues As Double(), _  
    endlimitValues As Double() _  
    ) As Integer
```

C#

```
public int SetLimitTestData(  
    string repeatedCapabilitiesID,  
    int[] limitLineType,  
    double[] beginLimitArgument,  
    double[] endLimitArgument,  
    double[] beginlimitValues,  
    double[] endlimitValues  
)
```

Visual C++

```
public:  
int SetLimitTestData(  

```

Visual C++

```
String repeatedCapabilitiesID,  
int limitLineType[],  
double beginLimitArgument[],  
double endLimitArgument[],  
double beginLimitValues[],  
double endLimitValues[]  
)
```

JavaScript

```
function setLimitTestData(repeatedCapabilitiesID, limitLineType,  
beginLimitArgument, endLimitArgument, beginLimitValues, endLimitValues);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

limitLineType

Type: **System.Int32** []

The type of the n–th limit line segment.

[AnalysisLimitLineOff](#)

[AnalysisLimitLineMax](#)

[AnalysisLimitLineMin](#)

[AnalysisLimitLineSingle](#)

beginLimitArgument

Type: **System.Double** []

The stimulus value in the start [point](#) of the n–th segment.

[endLimitArgument](#)

Type: **System.Double** []

The stimulus value in the end [point](#) of the n–th segment.

[beginLimitValues](#)

Type: **System.Double** []

The response value in the start [point](#) of the n–th segment.

[endLimitValues](#)

Type: **System.Double** []

The response value in the end [point](#) of the n–th segment.

Return Value

Success or failure code.

Back to [Methods](#)

SetPowerCalibrationTable

Description

Sets the power correction array (result of power calibration executed by PerformCalibrationAction function).

The array size is NOP, where NOP is the number of measurement points.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SetPowerCalibrationTable ( _  
    repeatedCapabilitiesID As String, _  
    powerValues As Double() _  
    ) As Integer
```

C#

```
public int SetPowerCalibrationTable(  
    string repeatedCapabilitiesID,  
    double[] powerValues  
)
```

Visual C++

```
public:  
int SetPowerCalibrationTable(  
    String repeatedCapabilitiesID,  
    double powerValues[]  
)
```

JavaScript

```
function setPowerCalibrationTable(repeatedCapabilitiesID, powerValues);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.

powerValues

Type: **System.Double** []

The array of the power correction value.

Return Value

Success or failure code.

Back to [Methods](#)

SetPowerLossCompensationTable

Description

Sets the loss compensation table used when the power calibration is executed by PerformCalibrationAction function.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SetPowerLossCompensationTable ( _  
    repeatedCapabilitiesID As String, _  
    frequencyValues As Double(), _  
    lossValues As Double() _  
    ) As Integer
```

C#

```
public int SetPowerLossCompensationTable(  
    string repeatedCapabilitiesID,  
    double[] frequencyValues,  
    double[] lossValues  
)
```

Visual C++

```
public:  
int SetPowerLossCompensationTable(  
    String repeatedCapabilitiesID,  
    double frequencyValues[],  
    double lossValues[]  
)
```

JavaScript

```
function setPowerLossCompensationTable(repeatedCapabilitiesID,  
frequencyValues, lossValues);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.

frequencyValues

Type: **System.Double** []

The array of value of frequency.

lossValues

Type: **System.Double** []

The array of value of the loss compensation (from –100 to +100 dB).

Return Value

Success or failure code.

Back to [Methods](#)

SetRippleLimitData

Description

Sets the data array, which is the limit line for the ripple limit function.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SetRippleLimitData ( _  
    repeatedCapabilitiesID As String, _  
    rippleLimitType As Integer(), _  
    beginLimitArgument As Double(), _  
    endLimitArgument As Double(), _  
    limitValues As Double() _  
    ) As Integer
```

C#

```
public int SetRippleLimitData(  
    string repeatedCapabilitiesID,  
    int[] rippleLimitType,  
    double[] beginLimitArgument,  
    double[] endLimitArgument,  
    double[] limitValues  
)
```

Visual C++

```
public:  
int SetRippleLimitData(  
    String repeatedCapabilitiesID,  
    int rippleLimitType[],
```

Visual C++

```
double beginLimitArgument[],  
double endLimitArgument[],  
double limitValues[]  
)
```

JavaScript

```
function setRippleLimitData(repeatedCapabilitiesID, rippleLimitType,  
beginLimitArgument, endLimitArgument, limitValues);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

rippleLimitType

Type: **System.Int32** []

The type of the n–th limit line segment.

[AnalysisRippleLimitOff](#)

[AnalysisRippleLimitOn](#)

beginLimitArgument

Type: **System.Double** []

The stimulus value in the beginning pointof the n–th segment.

endLimitArgument

Type: **System.Double** []

The stimulus value in the end point of the n -th segment.

limitValues

Type: **System.Double** []

The ripple limit value of the n -th segment.

Return Value

Success or failure code.

Back to [Methods](#)

SetSegmentData

Description

Sets the array of the segment sweep table.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SetSegmentData ( _  
    repeatedCapabilitiesID As String, _  
    startFrequencies As Double(), _  
    stopFrequencies As Double(), _  
    numberPoints As Integer(), _  
    listIFBandwidth As Boolean, _  
    ifBandwidthValues As Double(), _  
    listPower As Boolean, _  
    powerValues As Double(), _  
    listDelay As Boolean, _  
    delayValues As Double(), _  
    listTime As Boolean, _  
    timeValues As Double() _  
) As Integer
```

C#

```
public int SetSegmentData(  
    string repeatedCapabilitiesID,  
    double[] startFrequencies,  
    double[] stopFrequencies,  
    int[] numberPoints,  
    bool listIFBandwidth,
```

C#

```
double[] ifBandwidthValues,  
bool listPower,  
double[] powerValues,  
bool listDelay,  
double[] delayValues,  
bool listTime,  
double[] timeValues  
)
```

Visual C++

```
public:  
int SetSegmentData(  
    String repeatedCapabilitiesID,  
    double startFrequencies[],  
    double stopFrequencies[],  
    int numberPoints[],  
    bool listIFBandwidth,  
    double ifBandwidthValues[],  
    bool listPower,  
    double powerValues[],  
    bool listDelay,  
    double delayValues[],  
    bool listTime,  
    double timeValues[]  
)
```

JavaScript

```
function setSegmentData(repeatedCapabilitiesID, startFrequencies,  
stopFrequencies, numberPoints, listIFBandwidth, ifBandwidthValues,  
listPower, powerValues, listDelay, delayValues, listTime, timeValues);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

startFrequencies

Type: **System.Double** []

The array of start values.

stopFrequencies

Type: **System.Double** []

The array of stop values.

numberPoints

Type: **System.Int32** []

The array of number of points.

listIFBandwidth

Type: **System.Boolean**

Setting of ifbwValues (0 – disabled, 1 – enabled).

ifBandwidthValues

Type: **System.Double** []

The array of value of IF bandwidth (if enabled).

listPower

Type: **System.Boolean**

Setting of *powerValues* (0 – disabled, 1 – enabled).

powerValues

Type: **System.Double []**

The array of value of power (if enabled).

listDelay

Type: **System.Boolean**

Setting of *delayValues* (0 – disabled, 1 – enabled).

delayValues

Type: **System.Double []**

The array of value of measurement delay (if enabled).

listTime

Type: **System.Boolean**

Setting of *timeValues* (0 – disabled, 1 – enabled).

timeValues

Type: **System.Double []**

Reserved for future use (if enabled).

Return Value

Success or failure code.

Back to [Methods](#)

SetSubclassStdOrder

Description

Sets the subclass used to specify classes of calibration standards by functions:

[SetCalKitOrderOpen](#)

[SetCalKitOrderShort](#)

[SetCalKitOrderLoad](#)

[SetCalKitOrderThru](#)

[SetCalKitOrderTRLLine](#)

[SetCalKitOrderTRLThru](#)

[SetCalKitOrderTRLReflect](#)

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SetSubclassStdOrder ( _  
    subclassNumber As Integer _  
) As Integer
```

C#

```
public int SetSubclassStdOrder(  
    int subclassNumber  
)
```

Visual C++

```
public:  
int SetSubclassStdOrder(  
    int subclassNumber
```

Visual C++

```
)
```

JavaScript

```
function setSubclassStdOrder(subclassNumber);
```

Parameters

subclassNumber

Type: **System.Int32**

The subclass number of the calibration standard.

Return Value

Success or failure code.

Back to [Methods](#)

SetTouchstoneFileType

Description

Sets the Touchstone file type and the port numbers, when saving S-parameters by [SaveTouchstoneFile\(String, String\)](#) function.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SetTouchstoneFileType ( _  
    repeatedCapabilitiesID As String, _  
    touchstoneType As Integer, _  
    ports As Integer() _  
) As Integer
```

C#

```
public int SetTouchstoneFileType(  
    string repeatedCapabilitiesID,  
    int touchstoneType,  
    int[] ports  
)
```

Visual C++

```
public:  
int SetTouchstoneFileType(  
    String repeatedCapabilitiesID,  
    int touchstoneType,  
    int ports[]  
)
```

JavaScript

```
function setTouchstoneFileType(repeatedCapabilitiesID, touchstoneType,  
ports);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

touchstoneType

Type: **System.Int32**

The type of Touchstone file:

[SaveTouchstoneFileS1p](#)

[SaveTouchstoneFileS2p](#)

[SaveTouchstoneFileS3p](#)

[SaveTouchstoneFileS4p](#)

ports

Type: **System.Int32 []**

The array of port numbers. The number of ports must match *touchstoneType*.

Return Value

Success or failure code.

Back to [Methods](#)

SystemHide

Description

Minimizes the analyzer main window removing it from the desktop.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SystemHide As Integer
```

C#

```
public int SystemHide()
```

Visual C++

```
public:  
int SystemHide()
```

JavaScript

```
function systemHide();
```

Return Value

Success or failure code.

Back to [Methods](#)

SystemPreset

Description

Resets the Analyzer to the factory settings.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SystemPreset As Integer
```

C#

```
public int SystemPreset()
```

Visual C++

```
public:  
int SystemPreset()
```

JavaScript

```
function systemPreset();
```

Return Value

Success or failure code.

Back to [Methods](#)

SystemShow

Description

Restores the analyzer main window hidden by [SystemHide](#) function.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function SystemShow As Integer
```

C#

```
public int SystemShow()
```

Visual C++

```
public:  
int SystemShow()
```

JavaScript

```
function systemShow();
```

Return Value

Success or failure code.

Back to [Methods](#)

TakePowerCalSweep

Description

Measures the power calibration data for the port using the power meter controlled via USB or USB/GPIB.

Calculates calibration coefficients on completion of the measurement, and turns ON the power correction for the port.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function TakePowerCalSweep ( _  
    repeatedCapabilitiesID As String _  
) As Integer
```

C#

```
public int TakePowerCalSweep(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int TakePowerCalSweep(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function takePowerCalSweep(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.

Return Value

Success or failure code.

Back to [Methods](#)

TestBeepComplete

Description

Generates a beep to notify of the completion of the operation.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function TestBeepComplete As Integer
```

C#

```
public int TestBeepComplete()
```

Visual C++

```
public:  
int TestBeepComplete()
```

JavaScript

```
function testBeepComplete();
```

Return Value

Success or failure code.

Back to [Methods](#)

TestBeepWarning

Description

Generates a beep to notify of warning.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function TestBeepWarning As Integer
```

C#

```
public int TestBeepWarning()
```

Visual C++

```
public:  
int TestBeepWarning()
```

JavaScript

```
function testBeepWarning();
```

Return Value

Success or failure code.

Back to [Methods](#)

TimeDomainSetFrequencyLowPass

Description

Changes the frequency range to match with the low-pass type of the time domain transformation function.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function TimeDomainSetFrequencyLowPass ( _  
    repeatedCapabilitiesID As String _  
) As Integer
```

C#

```
public int TimeDomainSetFrequencyLowPass(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int TimeDomainSetFrequencyLowPass(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function timeDomainSetFrequencyLowPass(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

Return Value

Success or failure code.

Back to [Methods](#)

TraceHoldRestart

Description

This command resets the trace hold function.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function TraceHoldRestart ( _  
    repeatedCapabilitiesID As String _  
) As Integer
```

C#

```
public int TraceHoldRestart(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int TraceHoldRestart(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function traceHoldRestart(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

Return Value

Success or failure code.

Back to [Methods](#)

TriggerImmediate

Description

Generates a trigger signal and initiates a sweep under the following conditions: - Trigger source is set to the BUS (set by the command TRIG:SOUR BUS), otherwise an error occurs and the command is ignored. - Analyzer must be in the trigger waiting state, otherwise (the analyzer is in the measurement state or in the hold state) an error occurs and the function is ignored. The function is completed immediately after the generation of the trigger signal (does not wait the end of a sweep).

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function TriggerImmediate ( _  
    repeatedCapabilitiesID As String _  
) As Integer
```

C#

```
public int TriggerImmediate(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int TriggerImmediate(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function triggerImmediate(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

Pass VI_NULL or empty string if operation does not apply to a repeated capability.

Return Value

Success or failure code.

Back to [Methods](#)

TriggerInit

Description

Puts the channel to the Trigger Waiting state for the one trigger event.

The channel should be in the hold state, otherwise an error occurs and the command is ignored.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function TriggerInit ( _  
    repeatedCapabilitiesID As String _  
) As Integer
```

C#

```
public int TriggerInit(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int TriggerInit(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function triggerInit(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Return Value

Success or failure code.

Back to [Methods](#)

TriggerRestart

Description

Aborts the sweep. The channels in the Single trigger initiation mode transit to the Hold state.

The channels in the Continuous trigger initiation mode transit to the trigger waiting state, if the trigger source is set to Internal, the channel immediately starts a new sweep.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function TriggerRestart ( _  
    repeatedCapabilitiesID As String _  
) As Integer
```

C#

```
public int TriggerRestart(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int TriggerRestart(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function triggerRestart(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

Pass VI_NULL or empty string if operation does not apply to a repeated capability.

Return Value

Success or failure code.

Back to [Methods](#)

TriggerSingleAndWait

Description

Generates a trigger signal and initiates a sweep under the following conditions: - Trigger source is set to the BUS (set by the CMTNA_ATTR_STIMULUS_TRIGGER_SOURCE), otherwise an error occurs and the function is ignored. - Analyzer must be in the trigger waiting state, otherwise (the analyzer is in the measurement state or in the hold state) an error occurs and the function is ignored.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function TriggerSingleAndWait ( _  
    repeatedCapabilitiesID As String, _  
    timeoutMilliseconds As Integer _  
) As Integer
```

C#

```
public int TriggerSingleAndWait(  
    string repeatedCapabilitiesID,  
    int timeoutMilliseconds  
)
```

Visual C++

```
public:  
int TriggerSingleAndWait(  
    String repeatedCapabilitiesID,  
    int timeoutMilliseconds  
)
```

JavaScript

```
function triggerSingleAndWait(repeatedCapabilitiesID, timeoutMilliseconds);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

Pass VI_NULL or empty string if operation does not apply to a repeated capability.

timeoutMilliseconds

Type: **System.Int32**

Maximum time out. This value is expressed in milliseconds.

Return Value

Success or failure code.

Back to [Methods](#)

TriggerSingle

Description

Generates a trigger signal and initiates a sweep under the following conditions: - Trigger source is set to the BUS (set by the CMTNA_ATTR_STIMULUS_TRIGGER_SOURCE), otherwise an error occurs and the function is ignored. - Analyzer must be in the trigger waiting state, otherwise (the analyzer is in the measurement state or in the hold state) an error occurs and the function is ignored.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function TriggerSingle ( _  
    repeatedCapabilitiesID As String _  
) As Integer
```

C#

```
public int TriggerSingle(  
    string repeatedCapabilitiesID  
)
```

Visual C++

```
public:  
int TriggerSingle(  
    String repeatedCapabilitiesID  
)
```

JavaScript

```
function triggerSingle(repeatedCapabilitiesID);
```

Parameters

repeatedCapabilitiesID

Type: **System.String**

Pass VI_NULL or empty string if operation does not apply to a repeated capability.

Return Value

Success or failure code.

Back to [Methods](#)

UnlockSession

Description

Releases a lock obtained on an device driver session by calling the CmtNA_LockSession function.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function UnlockSession ( _  
    <OutAttribute> ByRef callerHasLock As Boolean _  
    ) As Integer
```

C#

```
public int UnlockSession(  
    out bool callerHasLock  
)
```

Visual C++

```
public:  
int UnlockSession(  
    bool *callerHasLock  
)
```

JavaScript

```
function unlockSession(callerHasLock);
```

Parameters

callerHasLock

Type: **System.Boolean**

Return Value

Success or failure code.

Back to [Methods](#)

WaitForOperationComplete

Description

Method returns when all pending operations are complete or `maxTimeMilliseconds` exceeded.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function WaitForOperationComplete ( _  
    timeoutMilliseconds As Integer _  
) As Integer
```

C#

```
public int WaitForOperationComplete(  
    int timeoutMilliseconds  
)
```

Visual C++

```
public:  
int WaitForOperationComplete(  
    int timeoutMilliseconds  
)
```

JavaScript

```
function waitForOperationComplete(timeoutMilliseconds);
```

Parameters

timeoutMilliseconds

Type: **System.Int32**

Maximum time out. This value is expressed in milliseconds.

Return Value

Success or failure code.

Back to [Methods](#)

WaitForSweepComplete

Description

Wait the end of the sweep.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function WaitForSweepComplete ( _  
    timeoutMilliseconds As Integer _  
) As Integer
```

C#

```
public int WaitForSweepComplete(  
    int timeoutMilliseconds  
)
```

Visual C++

```
public:  
    int WaitForSweepComplete(  
        int timeoutMilliseconds  
    )
```

JavaScript

```
function waitForSweepComplete(timeoutMilliseconds);
```

Parameters

timeoutMilliseconds

Type: **System.Int32**

Maximum time out. This value is expressed in milliseconds.

Return Value

Success or failure code.

Back to [Methods](#)

WaitForTriggerState

Description

Waits the specified state of the analyzer has been reached.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Function WaitForTriggerState ( _  
    state As Integer, _  
    timeoutMilliseconds As Integer _  
) As Integer
```

C#

```
public int WaitForTriggerState(  
    int state,  
    int timeoutMilliseconds  
)
```

Visual C++

```
public:  
int WaitForTriggerState(  
    int state,  
    int timeoutMilliseconds  
)
```

JavaScript

```
function waitForTriggerState(state, timeoutMilliseconds);
```

Parameters

state

Type: **System.Int32**

Specifies state of the analyzer.

timeoutMilliseconds

Type: **System.Int32**



The time to wait for complete operation before returning an error.


Return Value

Success or failure code.



Back to [Methods](#)



class Attributes


	Name	Description
	AdapterRemovalCutoffFreq	<p>Sets/Gets the value of the cutoff frequency of the waveguide adapter.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	AdapterRemovalDelay	<p>Sets/Gets the approximate delay value of an adapter in the adapter removal/insertion function.</p> <p>This value is used to eliminate the uncertainty of ± 180 deg when calculating the phase response of the adapter.</p> <p>The sign of the value depends on the type of the removal / insertion function.</p> <p>The value must be negative for the adapter removal function and must be positive for the adapter insertion function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p>



	Name	Description
		<p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	AdapterRemovalLength	<p>Sets/Gets the approximate value of the mechanical length of the adapter in the adapter removal/insertion function.</p> <p>This value is used to eliminate the uncertainty of ± 180 deg when calculating the phase response of the adapter.</p> <p>The sign of the value depends on the type of the removal / insertion function.</p> <p>The value must be negative for the adapter removal function and must be positive for the adapter insertion function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>




	Name	Description
	AdapterRemovalMedia	<p>Sets/Gets the adapter media in the adapter removal/insertion function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>Field Value</p> <p>AdapterRemovalMediaCoaxial</p> <p>AdapterRemovalMediaWaveguide</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	AdapterRemovalPermittivity	<p>Sets/Gets the value of the permittivity of an adapter media in the adapter removal/insertion function.</p> <p>When setting the adapter length, this parameter is used to calculate the adapter delay; therefore this parameter must be set before setting of the adapter length. When setting the adapter delay, this parameter is not used.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p>



	Name	Description
		<p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	AdapterRemovalPort	<p>Sets the port number and sets the adapter removal/insertion function for the calculation of the calibration coefficients when ApplyCalibration(String) function has been executed.</p> <p>Used as attributeID with function SetAttributeVInt32</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	AdapterRemovalUnit	<p>Sets/Gets the display units of the adapter delay (length) in the adapter removal/insertion function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeVInt32</p> <p>GetAttributeVInt32</p> <p>Field Value</p> <p>AdapterRemovalUnitMeters</p> <p>AdapterRemovalUnitSeconds</p> <p>The repeated capability identifier:</p>



	Name	Description
		The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.
	AnalysisConversion	<p>Turns ON/OFF the S-parameter conversion function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	AnalysisConversionFunction	<p>Sets/Gets the S-parameter conversion function type.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>Field Value</p> <p>ConversionFunctionConjugation</p>



	Name	Description
		<p>ConversionFunctionSInverse</p> <p>ConversionFunctionYReflection</p> <p>ConversionFunctionYTransmission</p> <p>ConversionFunctionYTransShunt</p> <p>ConversionFunctionZReflection</p> <p>ConversionFunctionZTransmission</p> <p>ConversionFunctionZTransShunt</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	AnalysisDeembedding	<p>Turns ON/OFF the 2-port network de-embedding function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p>

	Name	Description
		The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.
	AnalysisDeembeddingPort	<p>Turns ON/OFF the 2-port network de-embedding function for specified port.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.</p>
	AnalysisDeembeddingPortFile	<p>Sets/Gets the name of *.s2p file of the de-embedded circuit of the 2-port network de-embedding function.</p> <p>The file contains the circuit S-parameters in Touchstone format.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViString</p> <p>GetAttributeViString</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding</p>



	Name	Description
		Channel index and Port index i.e. "Channel1:Port2" and so on.
	AnalysisEmbedding	<p>Turns ON/OFF the 2-port network embedding function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	AnalysisEmbeddingPort	<p>Turns ON/OFF the 2-port network embedding function for each port.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.</p>
	AnalysisEmbeddingPortFile	Sets/Gets the name of *.s2p file of the embedded circuit of the 2-port network embedding function.


	Name	Description
		<p>The file contains the circuit S-parameters in Touchstone format.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViString</p> <p>GetAttributeViString</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.</p>
	AnalysisFixtureSimulator	<p>Turns ON/OFF the fixture simulation function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	AnalysisLimitLineDisplay	<p>Turns ON/OFF the limit line display of the limit test function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p>



	Name	Description
		<p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	AnalysisLimitTest	<p>Turns ON/OFF the limit test.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	AnalysisLimitTestFailSign	<p>Turns ON/OFF the "Fail" sign display, when performing limit test.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p>


	Name	Description
		<p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	AnalysisLimitTestPoints	<p>Gets the number of the measurement points that failed the limit test.</p> <p>The stimulus data array of these points can be read out by GetLimitTestReport function.</p> <p>Used as attributeID with function GetAttributeViInt32</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	AnalysisPortZConversion	<p>Turns ON/OFF the port impedance conversion function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p>



	Name	Description
		<p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	AnalysisPortZConversionImag	<p>Sets/ Gets the imaginary part of the impedance of the port impedance conversion function. The value is expressed in Ohm.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.</p>
	AnalysisPortZConversionReal	<p>Sets/ Gets the real part of the impedance of the port impedance conversion function. The value is expressed in Ohm.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding</p>



	Name	Description
		Channel index and Port index i.e. "Channel1:Port2" and so on.
	AnalysisPortZConversionZ0	<p>Sets/Gets the value of the impedance for port impedance conversion function. The value is expressed in Ohm.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.</p>
	AnalysisRippleLimitDisplay	<p>Turns ON/OFF the ripple limit line display.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e.</p>



	Name	Description
		"Channel1:Measurement3" and so on.
	AnalysisRippleLimitFailSign	<p>Turns ON/OFF the "Fail" sign display for ripple limit test.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	AnalysisRippleTest	<p>Turns ON/OFF the ripple limit test.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>



	Name	Description
	AnalysisRippleValueType	<p>Sets/Gets the display type of the ripple value in the specified band.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>Field Value</p> <p>AnalysisRippleValueAbsolute</p> <p>AnalysisRippleValueMargin</p> <p>AnalysisRippleValueOff</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	AnalysisTimeDomain	<p>Turns ON/OFF the time domain transformation function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p>



	Name	Description
		<p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	AnalysisTimeDomainReflectionType	<p>Sets/Gets the reflection distance either one way or round trip for the time domain transformation function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>Field Value</p> <p>TimeDomainReflectionOneWay</p> <p>TimeDomainReflectionRoundTrip</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e.</p>



	Name	Description
		"Channel1:Measurement3" and so on.
	AnalysisTimeDomainUnits	<p>Sets/Gets the transformation unit for the time domain transformation function: seconds, meters, feet.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>Field Value</p> <p>TimeDomainUnitsFeet</p> <p>TimeDomainUnitsMeters</p> <p>TimeDomainUnitsSeconds</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	AttrPrintColor	<p>Sets/Gets the color chart for the image printout.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p>

	Name	Description
		GetAttributeViInt32 Field Value PrintBlackAndWhite PrintColor PrintGrayScale The repeated capability identifier: Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.
	AutocalAutoOrientation	Turns ON/OFF the Auto-Orientation function when the AutoCal Module calibration is executed. Used as attributeID with functions: SetAttributeViBoolean GetAttributeViBoolean The repeated capability identifier: Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.
	AutocalCharacterization	Sets/Gets the characterization number used when executing AutoCal (factory or user characterizations). Used as attributeID with functions:



	Name	Description
		SetAttributeViInt32 GetAttributeViInt32 Field Value AutocalCharacterizationFactory AutocalCharacterizationUser1 AutocalCharacterizationUser2 AutocalCharacterizationUser3 The repeated capability identifier: Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.
	AutocalModuleReady	Gets the ready state of the AutoCal module. The state is True when the AutoCal module is connected. Used as attributeID with function GetAttributeViBoolean The repeated capability identifier: Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.
	AutocalOrientationPort	Sets/Gets the AutoCal module port number which is connected to a specified port of Network Analyzer. Used as attributeID with functions: SetAttributeViInt32


	Name	Description
		<p>GetAttributeViInt32</p> <p>Field Value</p> <p>AutocalOrientationPortA</p> <p>AutocalOrientationPortB</p> <p>AutocalOrientationPortC</p> <p>AutocalOrientationPortD</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	AutocalTemperature	<p>Gets the temperature of the AutoCal module connected to the Analyzer. The value is expressed in Celsius degrees.</p> <p>Used as attributeID with functions: GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	AutocalUnknownThru	<p>Turns ON/OFF the Unknown Thru feature when the AutoCal Module calibration is executed.</p> <p>Used as attributeID with functions: SetAttributeViBoolean</p>


	Name	Description
		GetAttributeViBoolean The repeated capability identifier: Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.
	AutoPortExtensionAdjustMismatch	Turns ON/OFF the usage of "Loss at DC" value for the results of the auto port extension function. Used as attributeID with functions: SetAttributeViBoolean GetAttributeViBoolean The repeated capability identifier: The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.
	AutoPortExtensionIncludeLoss	Turns ON/OFF the usage of "Loss1" and "Loss2" values for the results of the auto port extension function. Used as attributeID with functions: SetAttributeViBoolean GetAttributeViBoolean The repeated capability identifier: The physical names are supported along with the corresponding



	Name	Description
		Channel index i.e. "Channel1" and so on.
	AutoPortExtensionMethod	<p>Sets/Gets the frequency range used for calculation of the results of the Auto Port Extension function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>Field Value</p> <p>AutoPortExtensionActiveMarker</p> <p>AutoPortExtensionCurrentSpan</p> <p>AutoPortExtensionUserSpan</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	AutoPortExtensionUserSpanStart	<p>Sets/Gets the start value of the user span of the auto port extension function. The value is expressed in Hertz (Hz).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p>



	Name	Description
		The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.
	AutoPortExtensionUserSpanStop	<p>Sets/Gets the stop value of the user span of the auto port extension function. The value is expressed in Hertz (Hz).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	BeepCompleteOn	<p>Turns ON/OFF the beeper notifying of the completion of the operation.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	BeepWarning	Turns ON/OFF the beeper notifying of warning.



	Name	Description
		<p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	CalibrationAutoSelectZ0	<p>Turns ON/OFF the auto-select Z0 function.</p> <p>When enabled the function sets the port impedance Z0 to the corresponding value of measuring calibration standard.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	CalibrationCorrection	<p>Turns ON/OFF the S-parameter error correction.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p>

	Name	Description
		<p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	CalibrationPortZ0	<p>Gets the system impedance Z0 or the the impedance Z0 of port. The value is expressed in Ohm.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string if is used the system impedance Z0. If id used the the impedance Z0 of port, the physical names are supported along with the corresponding Port index i.e. "Port1" and so on.</p>
	CalibrationTriggerSource	<p>Enables/Disables the internal trigger source for calibration.</p> <p>If the internal trigger source for calibration is enabled then a command of the calibration standard measurement starts the measurement immediately. If the internal trigger source for calibration is disabled then the system trigger source is used (which is set for regular measurement with StimulusTriggerSource to start the calibration standard measurement.</p>

	Name	Description
		<p>The system trigger source also enables the averaging trigger MeasurementAvgTrigger and the point trigger StimulusExtTriggerEvent for calibration.</p> <p>NOTE: When the system trigger source is selected you should avoid the program trigger source (BUS), otherwise the program deadlock is possible.</p> <p>Note: The command isn't applied to the electronic calibration, the power calibration and the receiver calibration. The internal trigger is always used in these cases.</p> <p>Field Value</p> <p>CalibrationTriggerSourceInternal</p> <p>CalibrationTriggerSourceSystem</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	CalibrationType	<p>Gets the calibration type for the calculation of the calibration coefficients on completion of the calibration executed by ApplyCalibration function.</p> <p>Used as attributeID with function GetAttributeVInt32.</p> <p>Field Value</p>

	Name	Description
		CalibrationType1path2port CalibrationType1portSol CalibrationType2portSolt CalibrationType2portTrl CalibrationTypeNotDefined CalibrationTypeResponseOpen CalibrationTypeResponseShort CalibrationTypeResponseThru <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	CalkitDescription	<p>Sets/Gets the calibration kit description string.</p> <p>Used as attributeID with functions:</p> SetAttributeViString GetAttributeViString <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	CalkitLabel	Sets/Gets the calibration kit label.


	Name	Description
		<p>Used as attributeID with functions:</p> <p>SetAttributeViString</p> <p>GetAttributeViString</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	CalkitSelected	<p>Sets/Gets the number of the selected calibration kit in the table of calibration kits.</p> <p>The selected calibration kit is used in the subsequent calibration and is used for editing by CalkitStandardOpen, CalkitStandardShort e.c.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	CalkitStandardArbitrary	<p>Sets/Gets the value of the arbitrary impedance for the load standard. The value is expressed in Ohm.</p> <p>Used as attributeID with functions:</p>




	Name	Description
		SetAttributeViReal64 GetAttributeViReal64 The repeated capability identifier: The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.
	CalkitStandardC0	Sets/Gets the C0 value for the open calibration standard. The value is expressed in 1E-15 F. Used as attributeID with functions: SetAttributeViReal64 GetAttributeViReal64 The repeated capability identifier: The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.
	CalkitStandardC1	Sets/Gets the C1 value for the open calibration standard. The value is expressed in 1E-27 F/Hz. Used as attributeID with functions: SetAttributeViReal64 GetAttributeViReal64 The repeated capability identifier:


	Name	Description
		The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.
	CalkitStandardC2	<p>Sets/Gets the C2 value for the open calibration standard. The value is expressed in $1\text{E}-36 \text{ F/Hz}^2$.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.</p>
	CalkitStandardC3	<p>Sets/Gets the C3 value for the open calibration standard. The value is expressed in $1\text{E}-45 \text{ F/Hz}^3$.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.</p>



	Name	Description
	CalkitStandardL0	<p>Sets/Gets the L0 value for the short calibration standard. The value is expressed in $1\text{E}-12\text{ H}$.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.</p>
	CalkitStandardL1	<p>Sets/Gets the L1 value for the short calibration standard. The value is expressed in $1\text{E}-24\text{ H/Hz}$.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.</p>
	CalkitStandardL2	<p>Sets/Gets the L2 value for the short calibration standard. The value is expressed in $1\text{E}-33\text{ H/Hz}^2$.</p> <p>Used as attributeID with functions:</p>

	Name	Description
		SetAttributeViReal64 GetAttributeViReal64 The repeated capability identifier: The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.
	CalkitStandardL3	Sets/Gets the L3 value for the short calibration standard. The value is expressed in $1\text{E}-42 \text{ H/Hz}^3$. Used as attributeID with functions: SetAttributeViReal64 GetAttributeViReal64 The repeated capability identifier: The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.
	CalkitStandardLabel	Sets/Gets the label for the calibration standard. Used as attributeID with functions: SetAttributeViString GetAttributeViString The repeated capability identifier:




	Name	Description
		The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.
	CalkitStandardMaxFrequency	<p>Sets/Gets the maximum frequency limit of the calibration standard. The value is expressed in hertz (Hz).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.</p>
	CalkitStandardMinFrequency	<p>Sets/Gets the minimum frequency limit of the calibration standard. The value is expressed in hertz (Hz).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.</p>

	Name	Description
	CalkitStandardOffsetDelay	<p>Sets/Gets the offset delay value for the calibration standard. The value is expressed in seconds (sec).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.</p>
	CalkitStandardOffsetLoss	<p>Sets/Gets the offset loss value for the calibration standard. The value is expressed in Ohm/seconds.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.</p>
	CalkitStandardOffsetZ0	<p>Sets/Gets the offset Z0 value for the calibration standard. The value is expressed in Ohm.</p> <p>Used as attributeID with functions:</p>



	Name	Description
		SetAttributeViReal64 GetAttributeViReal64 The repeated capability identifier: The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.
	CalkitStandardType	Sets/Gets the type of calibration standard. Used as attributeID with functions: SetAttributeViInt32 GetAttributeViInt32 Field Value CalkitStandardDataBased CalkitStandardLoad CalkitStandardNone CalkitStandardOpen CalkitStandardShort CalkitStandardSlidingLoad CalkitStandardThruDelay CalkitStandardUnknThru The repeated capability identifier:



	Name	Description
		The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.
	CorrectionState	<p>Gets the interpolation/extrapolation status of the error correction.</p> <p>Used as attributeID with function GetAttributeViString</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	CorrectionStatus	<p>Gets the interpolation/extrapolation status of the error correction.</p> <p>Used as attributeID with function GetAttributeViString</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e.</p>



	Name	Description
		"Channel1:Measurement3" and so on.
	CorrectionType	<p>Gets the applied calibration type and port numbers for the specified trace.</p> <p>Used as attributeID with function GetAttributeViString</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index end Measurement index i.e. "Channel1:Measurement3" and so on.</p>
	DevicePxiChassis	<p>Gets the identifier for the PXI chassis in which the CMT vector network analyzer is installed.</p> <p>Used as attributeID with function GetAttributeViInt32</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	DevicePxiSlot	<p>Gets the number for the PXI slot in which the CMT vector network analyzer is installed.</p> <p>Used as attributeID with function GetAttributeViInt32</p>



	Name	Description
		Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.
	DeviceReady	<p>Gets the ready state of the Analyzer.</p> <p>The state is True when analyzer hardware is connected, powered and the boot process is completed (about 15 sec).</p> <p>Used as attributeID with function GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	DeviceSerialNumber	<p>Gets the instrument serial number.</p> <p>Used as attributeID with function GetAttributeViString</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	DeviceTemperature	<p>Gets the specified sensor temperature inside the Analyzer. The value is expressed in Celsius degrees.</p> <p>Used as attributeID with function GetAttributeViInt32</p> <p>The repeated capability identifier:</p>

	Name	Description
		Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.
	DisplayActiveChannel	<p>Sets/Gets the active channel.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>The repeated capability identifier:</p> <p>Pass VI_NULL or empty string if operation does not apply to a repeated capability. The physical names are supported along with the corresponding Channel index for the active trace number of the channel i.e. "Channel1" and so on.</p>
	DisplayActiveTrace	<p>Sets/Gets the active trace in channel.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace number of the channel i.e. "Channel1" and so on.</p>
	DisplayBackgroundColor	Sets/Gets the background color for trace display.



	Name	Description
		<p>The color is defined by its mix of red (RR), green (GG) and blue (BB) components each represented by 16-bit hexadecimal number.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeVInt32</p> <p>GetAttributeVInt32</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	DisplayChannelAllocation	<p>Sets/Gets the layout of the channel windows on the screen.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeVInt32</p> <p>GetAttributeVInt32</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	DisplayCycleTimeValue	<p>Gets the measured cycle time.</p> <p>The cycle time is the interval between the start of two adjacent sweeps. The cycle time is averaged by an exponential window with a time constant of about 0.5 sec. If the cycle time is</p>

	Name	Description
		<p>changed more than 100 usec in comparison with the averaged time, the averaging starts anew. The value is expressed in seconds (sec).</p> <p>Used as attributeID with functions: GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	DisplayDataTraceColor	<p>Sets/Gets the data trace color.</p> <p>The color is defined by its mix of red (RR), green (GG) and blue (BB) components each represented by 16-bit hexadecimal number.</p> <p>Used as attributeID with functions: SetAttributeViInt32 GetAttributeViInt32</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Measurement index i.e. "Measurement1" and so on.</p>
	DisplayGridColor	<p>Sets/Gets the grid and the graticule label color for trace display.</p>


	Name	Description
		<p>The color is defined by its mix of red (RR), green (GG) and blue (BB) components each represented by 16-bit hexadecimal number.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	DisplayInvertColor	<p>Turns ON/OFF the inversion of display colors of the traces area.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	DisplayMaximizeChannel	<p>Turns ON/OFF of the maximization of the active channel window.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p>


	Name	Description
		GetAttributeViBoolean The repeated capability identifier: Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.
	DisplayMaximizeTrace	Turns ON/OFF the active trace maximization inside the specified channel. Used as attributeID with functions: SetAttributeViBoolean GetAttributeViBoolean The repeated capability identifier: The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.
	DisplayMemoryTraceColor	Sets/Gets the memory trace color. The color is defined by its mix of red (RR), green (GG) and blue (BB) components each represented by 16-bit hexadecimal number. Used as attributeID with functions: SetAttributeViInt32 GetAttributeViInt32 The repeated capability identifier:

	Name	Description
		The physical names are supported along with the corresponding Measurement index i.e. "Measurement1" and so on.
	DisplaySystemDate	<p>Sets/Gets the current date.</p> <p>The date format: "YYYY,MM,DD" (YYYY - year, MM - month, DD - day).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	DisplaySystemTime	<p>Sets/Gets the current time.</p> <p>The time format: "hh,mm,ss" (hh - hour, mm - minute, ss - second).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViString</p> <p>GetAttributeViString</p> <p>The repeated capability identifier:</p>
	DisplayTitleData	<p>Sets/Gets the channel title label.</p> <p>Used as attributeID with functions:</p>

	Name	Description
		SetAttributeViBoolean GetAttributeViBoolean The repeated capability identifier: The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.
	DisplayTitleLabel	Turns ON/OFF the channel title display. Used as attributeID with functions: SetAttributeViBoolean GetAttributeViBoolean The repeated capability identifier: The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.
	DisplayTraceAllocation	Sets/Gets the layout of the graph in the channel window. Used as attributeID with functions: SetAttributeViInt32 GetAttributeViInt32 The repeated capability identifier: The physical names are supported along with the corresponding

	Name	Description
		Channel index i.e. "Channel1" and so on.
	DisplayTraceCount	<p>Sets/Gets the number of traces in the channel.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	DisplayTraceDataMath	<p>Sets/Gets the math operation between the data trace and the memory trace.</p> <p>The math result replaces the data trace. If the memory trace does not exist, the command is ignored.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>Field Value</p> <p>TraceDataMathAdd</p> <p>TraceDataMathDiv</p> <p>TraceDataMathMult</p>

	Name	Description
		TraceDataMathOff TraceDataMathSubt The repeated capability identifier: The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.
	DisplayTraceHoldType	Sets/Gets the type of the trace hold function. The function holds the trace at the maximum or minimum point. Used as attributeID with functions: SetAttributeViInt32 GetAttributeViInt32 Field Value TraceHoldMax TraceHoldMin TraceHoldOff The repeated capability identifier:

	Name	Description
		<p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	DisplayTraceType	<p>Turns ON/OFF the memory trace display.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeVInt32</p> <p>GetAttributeVInt32</p> <p>Field Value</p> <p>DisplayTraceData</p> <p>DisplayTraceDataAndMemory</p> <p>DisplayTraceMemory</p> <p>DisplayTraceOff</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Measurement index i.e. "Channel1:Measurement3" and so on.</p>
	DisplayUpdate	Turns ON/OFF the display update.



	Name	Description
		<p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	ExternalReferenceRoute	<p>Sets/Gets the route of the external 10 MHz reference frequency. (PXle-S5090 model only).</p> <p>The source of the reference ReferenceFrequencySource must be set to the EXternal.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>Field Value</p> <p>ReferenceRouteFront</p> <p>ReferenceRouteRear</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	ExternalTriggerRoute	<p>Sets/Gets the connector to use for the external trigger input in a PXI</p>



	Name	Description
		<p>system (command valid for PXle-S5090 model only).</p> <p>The trigger source must be set to the EXternal. One of the 10 routes can be selected.</p> <p>The same line cannot be selected as input and output trigger route.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>Field Value</p> <p>TriggerRouteSmb</p> <p>TriggerRoutePxiTrig0</p> <p>TriggerRoutePxiTrig1</p> <p>TriggerRoutePxiTrig2</p> <p>TriggerRoutePxiTrig3</p> <p>TriggerRoutePxiTrig4</p> <p>TriggerRoutePxiTrig5</p> <p>TriggerRoutePxiTrig6</p> <p>TriggerRoutePxiTrig7</p> <p>TriggerRouteStar</p> <p>The repeated capability identifier:</p>



	Name	Description
		Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.
	FrequencyDivider	<p>Sets/Gets the basic frequency range divisor of current port when the frequency offset feature is ON and offset type is "PORT".</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port1" and so on.</p>
	FrequencyMultiplier	<p>Sets/Gets the basic frequency range multiplier of port when the frequency offset feature is ON and offset type is "PORT".</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port1" and so on.</p>



	Name	Description
	FrequencyOffset	<p>Sets/Gets the basic frequency range offset of current port when the frequency offset feature is ON and offset type is "PORT".</p> <p>The value is expressed in hertz (Hz).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port1" and so on.</p>
	FrequencyOffsetStart	<p>Sets/Gets the frequency sweep start of current port when the frequency offset feature is ON and offset type is "PORT".</p> <p>The value is expressed in hertz (Hz).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port1" and so on.</p>


	Name	Description
	FrequencyOffsetStop	<p>Sets/Gets the frequency sweep stop of current port when the frequency offset feature is ON and offset type is "PORT".</p> <p>The value is expressed in hertz (Hz).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port1" and so on.</p>
	FrequencyOffsetType	<p>Sets/Gets the frequency offset type when the frequency offset feature is ON.</p> <p>There are two frequency offset types: "Port1/Port2" and "Source/Receivers". First offset type offsets ports against each other. Second offset type offsets source against receivers.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>Field Value</p> <p>FrequencyOffsetTypePortPort</p>



	Name	Description
		<p>FrequencyOffsetTypeSourceReceiver</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	FrequencyReceiverDivider	<p>Sets/Gets the basic frequency range divisor to get the receiver frequency when the frequency offset feature is ON and offset type is "SRCRCv".</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	FrequencyReceiverMultiplier	<p>Sets/Gets the basic frequency range multiplier to get the receiver frequency when the frequency offset feature is ON and offset type is "SRCRCv".</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p>


	Name	Description
		<p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	FrequencyReceiverOffset	<p>Sets/Gets the basic frequency range offset to get the receiver frequency when the frequency offset feature is ON and offset type is "SRCRCv".</p> <p>The value is expressed in hertz (Hz).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	FrequencyReceiverOffsetStart	<p>Sets/Gets the frequency sweep start of the receivers when the frequency offset feature is ON and offset type is "SRCRCv".</p> <p>The value is expressed in hertz (Hz).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p>

	Name	Description
		<p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	FrequencyReceiverOffsetStop	<p>Sets/Gets the frequency sweep stop of the receivers when the frequency offset feature is ON and offset type is "SRCRCv".</p> <p>The value is expressed in hertz (Hz).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	FrequencySourceDivider	<p>Sets/Gets the basic frequency range divisor to get the source frequency when the frequency offset feature is ON and offset type is "SRCRCv".</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p>

	Name	Description
		<p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	FrequencySourceMultiplier	<p>Sets/Gets the basic frequency range multiplier to get the source frequency when the frequency offset feature is ON and offset type is "SRCRCv".</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	FrequencySourceOffset	<p>Sets/Gets the basic frequency range offset to get the source frequency when the frequency offset feature is ON and offset type is "SRCRCv".</p> <p>The value is expressed in hertz (Hz).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p>

	Name	Description
		<p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	FrequencySourceOffsetStart	<p>Sets/Gets the frequency sweep start of the source when the frequency offset feature is ON and offset type is "SRCRCv".</p> <p>The value is expressed in hertz (Hz).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	FrequencySourceOffsetStop	<p>Sets/Gets the frequency sweep stop of the source when the frequency offset feature is ON and offset type is "SRCRCv".</p> <p>The value is expressed in hertz (Hz).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p>


	Name	Description
		<p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	FunctionDomainCoupling	<p>If the arbitrary range is turned ON by FunctionDomainState, it specifies whether all traces of repCapID use the same range (coupling) or each trace uses individual range when FunctionExecute(String) is executed.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	FunctionDomainStart	<p>Sets/Gets the start value of the analysis range of FunctionExecute(String) function.</p> <p>Used as attributeID with functions:</p>

	Name	Description
		<p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	FunctionDomainState	<p>Specifies whether an arbitrary range or the entire sweep range is used when FunctionExecute(String) is executed.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e.</p>



	Name	Description
		"Channel1:Measurement3" and so on.
	FunctionDomainStop	<p>Sets/Gets the stop value of the analysis range of FunctionExecute(String) function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	FunctionPeakExcursion	<p>Sets/Gets the lower limit for the peak excursion value when executing the peak search with FunctionExecute(String) function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p>



	Name	Description
		<p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	FunctionPeakPolarity	<p>Sets/Gets the polarity when performing the peak search with FunctionExecute(String) function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>Field Value</p> <p>FunctionPeakPolarityBoth</p> <p>FunctionPeakPolarityNegative</p> <p>FunctionPeakPolarityPositive</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e.</p>

	Name	Description
		"Channel1:Measurement3" and so on.
	FunctionPoints	<p>Gets the number of points (data pairs) of the analysis result by FunctionExecute(String) function.</p> <p>Always reads out 1, when the search is executed for the maximum, minimum, mean, standard deviation, peak, and peak-to-peak values.</p> <p>The actual number of points is read out, when the search is executed for all peak or all targets.</p> <p>Used as attributeID with functions: GetAttributeVInt32</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	FunctionTargetLevel	<p>Sets/Gets the target level when performing the search for the trace and the target level crosspoints with FunctionExecute(String) function.</p> <p>Used as attributeID with functions:</p>

	Name	Description
		<p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	FunctionTransitionType	<p>Sets/Gets the transition type when performing the search for the trace and the target level crosspoints with FunctionExecute(String) function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>Field Value</p> <p>FunctionTransitionTypeBoth</p> <p>FunctionTransitionTypeNegative</p> <p>FunctionTransitionTypePositive</p> <p>The repeated capability identifier:</p>

	Name	Description
		<p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	FunctionType	<p>Sets/Gets the type of analysis executed by FunctionExecute(String) function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>Field Value</p> <p>FunctionAllPeaks</p> <p>FunctionAllTarget</p> <p>FunctionMaximum</p> <p>FunctionMean</p> <p>FunctionMinimum</p> <p>FunctionPeak</p> <p>FunctionPeakToPeak</p> <p>FunctionStandardDeviation</p>


	Name	Description
		<p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	LimitLineResponseOffset	<p>Sets/Gets the value of the limit line offset along Y-axis.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	LimitLineStimulusOffset	<p>Sets/Gets the value of the limit line offset along X-axis.</p> <p>Used as attributeID with functions:</p>



	Name	Description
		<p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	LogicalName	<p>Logical Name identifies a driver session in the Configuration Store.</p> <p>If Logical Name is not empty, the driver was initialized from information in the driver session.</p> <p>If it is empty, the driver was initialized without using the Configuration Store.</p> <p>Used as attributeID with functions: GetAttributeViString</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	MarkerMathBandwidthSearch	<p>Turns ON/OFF the bandwidth search function.</p>



	Name	Description
		<p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	MarkerMathBandwidthSearchRef	<p>Selects the reference point for the bandwidth search function: reference marker or absolute maximum value of the trace.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e.</p>



	Name	Description
		"Channel1:Measurement3" and so on.
	MarkerMathBandwidthSearchType	<p>Sets/Gets the type of the bandwidth search function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>Field Value</p> <p>MarkerMathBandwidthSearchBandpass</p> <p>MarkerMathBandwidthSearchNotch</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	MarkerMathBandwidthSearchValue	<p>Sets/Gets the bandwidth definition value.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p>



	Name	Description
		<p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Marker index for the active trace of channel i.e. "Channel1:Marker2" and so on. The physical names are supported along with the corresponding Channel index and Measurement index and Marker index for the n–th trace of channel i.e. "Channel1:Measurement3:Marker 2" and so on.</p>
	MarkerMathFlatness	<p>Turns ON/OFF the marker FLATNESS function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	MarkerMathFlatnessStart	<p>Sets/Gets the number of the marker, which specifies the start</p>



	Name	Description
		<p>frequency of the FLATNESS function domain.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	MarkerMathFlatnessStop	<p>Sets/Gets the number of the marker, which specifies the stop frequency of the FLATNESS function domain.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and</p>



	Name	Description
		Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.
	MarkerMathStatisticRange	<p>Sets/Gets either partial frequency range or entire frequency range is used for math statistic calculation.</p> <p>The partial frequency range is limited by two markers.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	MarkerMathStatistics	<p>Turns ON/OFF the math statistics display.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p>


	Name	Description
		<p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	MarkerMathStatisticStart	<p>Sets/Gets the number of the marker, which specifies the start frequency of the math statistics range.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	MarkerMathStatisticStop	<p>Sets/Gets the number of the marker, which specifies the stop</p>

	Name	Description
		<p>frequency of the math statistics range.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	MarkerPropertiesAlign	<p>Sets/Gets the alignment mode of the marker display position of each trace, when the only active trace display feature is turned OFF by MarkerPropertiesMarkerActiveOnly.</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	MarkerPropertiesDataXPosition	<p>Sets/Gets the display position of the marker annotation on the X-axis by a percentage of the display width. The value is expressed in percentages (%).</p>


	Name	Description
		<p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Measurement index i.e. "Channel1:Measurement3" and so on.</p>
	MarkerPropertiesDataYPosition	<p>Sets/Gets the display position of the marker annotation on the Y-axis by a percentage of the display height.</p> <p>The value is expressed in percentages (%).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Measurement index i.e. "Channel1:Measurement3" and so on.</p>
	MarkerPropertiesDiscrete	<p>Turns ON/OFF the marker discrete mode.</p>



	Name	Description
		<p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	MarkerPropertiesMarkerActiveOnly	<p>Sets/Gets display either the active trace markers or all trace markers.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	MarkerPropertiesMarkerCouple	<p>Turns ON/OFF the marker coupling between traces.</p> <p>When coupled the markers of different traces with same number track the X-axis position.</p>

	Name	Description
		<p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	MarkerPropertiesMarkerTable	<p>Turns ON/OFF of the marker table.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	MarkersCount	<p>Sets\Gets the number of the turned ON markers.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p>



	Name	Description
		<p>GetAttributeViInt32</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	MarkerSearchCouple	<p>If the arbitrary search range is turned ON by MarkerSearchRange, specifies whether all traces of repCapID use the same range (coupling) or each trace uses individual range when the marker search is performed.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e.</p>


	Name	Description
		"Channel1:Measurement3" and so on.
	MarkerSearchPeakExcursion	<p>Sets/Gets the start value of the marker search range.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Marker index for the active trace of channel i.e. "Channel1:Marker2" and so on. The physical names are supported along with the corresponding Channel index and Measurement index and Marker index for the n–th trace of channel i.e. "Channel1:Measurement3:Marker 2" and so on.</p>
	MarkerSearchPeakPolarity	<p>Sets/Gets the peak polarity, when the marker search for peak is performed by MarkerFunctionExecute function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>Field Value</p> <p>MarkerSearchPeakPolarityBoth</p>



	Name	Description
		<p>MarkerSearchPeakPolarityNegative</p> <p>MarkerSearchPeakPolarityPositive</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Marker index for the active trace of channel i.e. "Channel1:Marker2" and so on.</p> <p>The physical names are supported along with the corresponding Channel index and Measurement index and Marker index for the n–th trace of channel i.e. "Channel1:Measurement3:Marker 2" and so on.</p>
	<p>MarkerSearchRange</p>	<p>Specifies whether an arbitrary range or the entire sweep range is used when the marker search is performed.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th</p>

	Name	Description
		trace of channel i.e. "Channel1:Measurement3" and so on.
	MarkerSearchStart	<p>Sets/Gets the start value of the marker search range.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	MarkerSearchStop	<p>Sets/Gets the stop value of the marker search range.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace</p>

	Name	Description
		<p>of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	MarkerSearchTargetTransition	<p>Sets/Gets the type of the target transition, when the marker search for transition is performed by MarkerFunctionExecute function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>Field Value</p> <p>MarkerSearchTargetTransitionBot h</p> <p>MarkerSearchTargetTransitionNeg ative</p> <p>MarkerSearchTargetTransitionPos itive</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Marker index for the active trace of channel i.e. "Channel1:Marker2" and so on. The physical names are supported along with the corresponding Channel index and Measurement index and Marker index for the n–th</p>

	Name	Description
		trace of channel i.e. "Channel1:Measurement3:Marker 2" and so on.
	MarkerSearchTargetValue	<p>Sets/Gets the target value, when the marker search for target is performed by MarkerFunctionExecute function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Marker index for the active trace of channel i.e. "Channel1:Marker2" and so on. The physical names are supported along with the corresponding Channel index and Measurement index and Marker index for the n-th trace of channel i.e. "Channel1:Measurement3:Marker 2" and so on.</p>
	MarkerSearchTracking	<p>Turns ON/OFF the marker search tracking.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p>

	Name	Description
		<p>The physical names are supported along with the corresponding Channel index and Marker index for the active trace of channel i.e. "Channel1:Marker2" and so on.</p> <p>The physical names are supported along with the corresponding Channel index and Measurement index and Marker index for the n–th trace of channel i.e. "Channel1:Measurement3:Marker 2" and so on.</p>
	MarkerSearchType	<p>Sets the type of the marker search, which is performed by MarkerFunctionExecute function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>Field Value</p> <p>MarkerSearchTypeMaximum</p> <p>MarkerSearchTypeMinimum</p> <p>MarkerSearchTypePeak</p> <p>MarkerSearchTypePeakLeft</p> <p>MarkerSearchTypePeakRight</p> <p>MarkerSearchTypeTarget</p> <p>MarkerSearchTypeTargetLeft</p> <p>MarkerSearchTypeTargetRight</p>

	Name	Description
		<p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Marker index for the active trace of channel i.e. "Channel1:Marker2" and so on.</p> <p>The physical names are supported along with the corresponding Channel index and Measurement index and Marker index for the n–th trace of channel i.e. "Channel1:Measurement3:Marker 2" and so on.</p>
	MarkerStimulus	<p>Sets/Gets the stimulus value of the marker.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Marker index for the active trace of channel i.e. "Channel1:Marker2" and so on.</p> <p>The physical names are supported along with the corresponding Channel index and Measurement index and Marker index for the n–th trace of channel i.e. "Channel1:Measurement3:Marker 2" and so on.</p>
	MeasurementAveraging	<p>Turns ON/OFF the measurement averaging function.</p>

	Name	Description
		<p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	MeasurementAveragingFactor	<p>Sets/Gets the averaging factor, when the averaging function is turned on.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	MeasurementAvgTrigger	<p>Turns ON/OFF the averaging trigger function.</p> <p>The function executes a sweep the number of times specified by the averaging factor with a single trigger for the channels with the averaging enabled.</p> <p>The averaging process begins again with each trigger.</p>

	Name	Description
		<p>Note: The point trigger function has priority against this command. When the point trigger is enabled the number of pulses equal to (number of points) x (averaging factor) is needed to complete the averaging.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	MeasurementDivisions	<p>Sets/Gets the number of the vertical scale divisions.</p> <p>For the rectangular format only.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	MeasurementElecDelay	<p>Sets/Gets the value of the electrical delay. The value is expressed in seconds (sec).</p>



	Name	Description
		<p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	MeasurementFormat	<p>Sets/Gets the display format specified by Measurement Format Enum for the measurement.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>Field Value</p> <p>MeasurementFormatExpandPhase</p> <p>MeasurementFormatGroupDelay</p> <p>MeasurementFormatImag</p> <p>MeasurementFormatLinMag</p>

	Name	Description
		<p> MeasurementFormatLogMag MeasurementFormatPhase MeasurementFormatPolarLin MeasurementFormatPolarLog MeasurementFormatPolarReallmage MeasurementFormatReal MeasurementFormatSmithAdmittance MeasurementFormatSmithComplex MeasurementFormatSmithLin MeasurementFormatSmithLog MeasurementFormatSmithReallmage MeasurementFormatSwr </p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>

	Name	Description
	MeasurementPhaseOffset	<p>Sets/Gets the value of the phase offset. The value is expressed in degrees.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	MeasurementRefPosition	<p>Sets/Gets the position of the reference line.</p> <p>For the rectangular format only.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Measurement index i.e.</p>




	Name	Description
		"Channel1:Measurement3" and so on.
	MeasurementRefValue	<p>Sets/Gets the value of the reference line (response value on the reference line).</p> <p>For the rectangular format only.</p> <p>The value is depending on the format.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Measurement index i.e. "Channel1:Measurement3" and so on.</p>
	MeasurementScaleDiv	<p>Sets/Gets the trace scale. Sets the scale per division, when the data format is the rectangular format.</p> <p>Sets the full scale value, when the data format is the Smith chart format or the polar format.</p> <p>The value is depending on the format.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p>



	Name	Description
		<p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Measurement index i.e. "Channel1:Measurement3" and so on.</p>
	MeasurementSmoothing	<p>Turns ON/OFF the trace smoothing.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	MeasurementSmoothingAperture	<p>Sets/Gets the smoothing aperture, when performing smoothing function. The value is expressed in percentages (%).</p> <p>Used as attributeID with functions:</p>



	Name	Description
		<p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	NumberOfPorts	<p>Gets the number of the ports.</p> <p>Used as attributeID with function GetAttributeViInt32</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	OutputTriggerRoute	<p>Sets/Gets the connector to use for the trigger output in a PXI system (command valid for PXIe-S5090 model only).</p> <p>One of the 9 routes can be selected. The same line cannot be selected as input and output trigger route.</p> <p>Used as attributeID with functions:</p>


	Name	Description
		SetAttributeViInt32 GetAttributeViInt32 Field Value TriggerRouteSmb TriggerRoutePxiTrig0 TriggerRoutePxiTrig1 TriggerRoutePxiTrig2 TriggerRoutePxiTrig3 TriggerRoutePxiTrig4 TriggerRoutePxiTrig5 TriggerRoutePxiTrig6 TriggerRoutePxiTrig7 The repeated capability identifier: Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.
	PortExtensions	Turns ON/OFF the port extension function. Used as attributeID with functions: SetAttributeViBoolean GetAttributeViBoolean


	Name	Description
		<p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	PortExtensionsFreq1Value	<p>Sets/Gets the values of the frequency 1 to calculate the loss for the port extension function. The value is expressed in Hertz (Hz).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.</p>
	PortExtensionsFreq2Value	<p>Sets/Gets the values of the frequency 2 to calculate the loss for the port extension function. The value is expressed in Hertz (Hz).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding</p>

	Name	Description
		Channel index and Port index i.e. "Channel1:Port2" and so on.
	PortExtensionsLdcValue	<p>Sets/Gets the loss value at DC for the port extension function. The value is expressed in decibels (dB).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.</p>
	PortExtensionsLoss1State	<p>Turns ON/OFF the loss compensation of the loss 1 for the port extension function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.</p>
	PortExtensionsLoss1Value	Sets/Gets the values of the loss 1 for the port extension function. The


	Name	Description
		<p>value is expressed in decibels (dB).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.</p>
	PortExtensionsLoss2State	<p>Turns ON/OFF the loss compensation of the loss 2 for the port extension function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.</p>
	PortExtensionsLoss2Value	<p>Sets/Gets the values of the loss 2 for the port extension function. The value is expressed in decibels (dB).</p> <p>Used as attributeID with functions:</p>



	Name	Description
		SetAttributeViReal64 GetAttributeViReal64 The repeated capability identifier: The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.
	PortExtensionsTime	Sets/Gets the electrical delay value for the port extension function. The value is expressed in seconds (sec). Used as attributeID with functions: SetAttributeViReal64 GetAttributeViReal64 The repeated capability identifier: The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.
	PowerCalibrationCorrection	Turns ON/OFF the power correction. Used as attributeID with functions: SetAttributeViBoolean GetAttributeViBoolean The repeated capability identifier:


	Name	Description
		The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.
	PowerCalibrationLossCompensation	<p>Turns ON/OFF the state of the loss compensation used when the power calibration is executed by TakePowerCalSweep(String) function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.</p>
	PowerSensorReady	<p>Gets the ready state of the power sensor.</p> <p>The state is True when the power sensor is ready.</p> <p>Used as attributeID with function GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	PowerSensorType	Sets/Gets the power sensor type to be used in a source power

	Name	Description
		<p>calibration.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>Field Value</p> <p>PowerSensorNrpxt</p> <p>PowerSensorNrpxz</p> <p>PowerSensorNrpxs</p> <p>PowerSensorU200x</p> <p>PowerSensorU848x</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	PowerTripAtOverload	<p>Turns ON/OFF the Power Trip at Overload function. Except for Planar-804/808/304 Models.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p>

	Name	Description
		Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.
	PrintDateAndTime	<p>Turns ON/OFF the date and time printout in the upper right corner of the image.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	PrintInvertImage	<p>Turns ON/OFF the inverted color image printout.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	ReferenceFrequencySource	<p>Sets/Gets the internal or external source of the reference frequency of 10 MHz.</p> <p>Used as attributeID with functions:</p>


	Name	Description
		SetAttributeViInt32 GetAttributeViInt32 Field Value ReferenceFrequencySourceExternal ReferenceFrequencySourceInternal The repeated capability identifier: Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.
	ReferenceMarker	Turns ON/OFF the reference marker. When the reference marker is turned ON, all the values of the other markers turn to relative values. Used as attributeID with functions: SetAttributeViBoolean GetAttributeViBoolean The repeated capability identifier: The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and

	Name	Description
		Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.
	SaveTouchstoneFileColumnSeparator	<p>Sets/Gets the Touchstone file separator symbol when the S–parameters are saved by SaveTouchstoneFile function.</p> <p>The attribute is used for the active channel.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>Field Value</p> <p>SaveTouchstoneFileColumnSeparatorSpace</p> <p>SaveTouchstoneFileColumnSeparatorTab</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	SaveTouchstoneFileFormat	<p>Sets/Gets the data format for the S–parameter saving by SaveTouchstoneFile function.</p> <p>The attribute is used for the active channel.</p> <p>Used as attributeID with functions:</p>

	Name	Description
		SetAttributeViInt32 GetAttributeViInt32 Field Value SaveTouchstoneFileFormatDbAngle SaveTouchstoneFileFormatMagnitudeAngle SaveTouchstoneFileFormatRealImage The repeated capability identifier: Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.
	SaveType	Sets/Gets the type of the Analyzer or channel state saving by SaveChannelToRegister function. Used as attributeID with functions: SetAttributeViInt32 GetAttributeViInt32 Field Value SaveTypeAll SaveTypeState SaveTypeStateAndCal SaveTypeStateAndCalAndMem



	Name	Description
		<p>SaveTypeStateAndTrace</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	SelectedMarker	<p>Sets the active marker.</p> <p>If the marker is not ON, this function will turn the marker ON. Turning ON a marker with the number from 1 to 15 will turn ON all the markers of smaller numbers. Turning ON the reference marker with number 16 does not turn ON the markers with the numbers from 1 to 15, but switches these markers to the relative measurement mode.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>




	Name	Description
	StimulusExtTriggerDelay	<p>Sets/Gets the response delay with respect to the external trigger signal. The value is expressed in seconds (sec).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	StimulusExtTriggerEvent	<p>Turns ON/OFF the point trigger feature for external trigger source.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>Field Value</p> <p>ExtTriggerEventOnSweep</p> <p>ExtTriggerEventOnPoint</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	StimulusExtTriggerPolarity	<p>Sets/Gets out the polarity of the external trigger.</p>



	Name	Description
		<p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>Field Value</p> <p>ExtTriggerPolarityNegative</p> <p>ExtTriggerPolarityPositive</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	StimulusExtTriggerPosition	<p>Sets/Gets the position of the external trigger.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>Field Value</p> <p>ExtTriggerPositionBsampling</p> <p>ExtTriggerPositionBsetup</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>




	Name	Description
	StimulusFrequencyCenter	<p>Sets/Gets the center value of the frequency sweep range. The value is expressed in hertz (Hz).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	StimulusFrequencyOffset	<p>Turns ON/OFF the frequency offset feature.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	StimulusFrequencySpan	<p>Sets/Gets the span value of the frequency sweep range. The value is expressed in hertz (Hz).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p>



	Name	Description
		GetAttributeViReal64 The repeated capability identifier: The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.
	StimulusFrequencyStart	Sets/Gets the start value of the frequency sweep range. The value is expressed in hertz (Hz). Used as attributeID with functions: SetAttributeViReal64 GetAttributeViReal64 The repeated capability identifier: The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.
	StimulusFrequencyStop	Sets/Gets the stop value of the frequency sweep range. The value is expressed in hertz (Hz). Used as attributeID with functions: SetAttributeViReal64 GetAttributeViReal64 The repeated capability identifier: The physical names are supported along with the corresponding



	Name	Description
		Channel index i.e. "Channel1" and so on.
	StimulusIfBandwidth	<p>Sets/Gets the bandwidth of the digital IF filter to be used in the measurement.</p> <p>IF Bandwidth is in Hz. The list of valid IF Bandwidths is different depending on the analyzer model.</p> <p>If an invalid number is specified, the analyzer will round up to the closest valid number.</p> <p>The value is expressed in hertz (Hz).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	StimulusMaxFrequency	<p>Gets the upper limit of the measurement frequency. The value is expressed in hertz (Hz).</p> <p>Used as attributeID with functions:</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p>

	Name	Description
		Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.
	StimulusMaxPoints	<p>Gets the maximum number of the measurement points.</p> <p>Used as attributeID with functions: GetAttributeViInt32</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	StimulusMinFrequency	<p>Gets the lower frequency of the measurement frequency. The value is expressed in hertz (Hz).</p> <p>Used as attributeID with functions: GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	StimulusOutputPower	<p>Sets/Gets the power level for the frequency sweep type. This value is expressed in decibels above 1 milliwatt (dBm).</p> <p>Used as attributeID with functions: SetAttributeViReal64 GetAttributeViReal64</p> <p>The repeated capability identifier:</p>



	Name	Description
		The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.
	StimulusOutputPowerPort	<p>Sets/Gets the power level of port for the frequency sweep type when the port couple feature is set to OFF.</p> <p>This value is expressed in decibels above 1 milliwatt (dBm).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port1" and so on.</p>
	StimulusOutputPowerPortCouple	<p>Turns ON/OFF the port power couple.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>


	Name	Description
	<u>StimulusOutputPowerRfout</u>	<p>Turns ON/OFF the RF signal output. Measurements cannot be performed when the RF signal output is turned OFF.</p> <p>Used as attributeID with functions:</p> <p><u>SetAttributeViBoolean</u></p> <p><u>GetAttributeViBoolean</u></p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	<u>StimulusOutputPowerSlope</u>	<p>Sets/Gets the power slope value for the frequency sweep. This value is expressed in decibels/gigahertz (dB/GHz).</p> <p>Used as attributeID with functions:</p> <p><u>SetAttributeViReal64</u></p> <p><u>GetAttributeViReal64</u></p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	<u>StimulusOutputPowerSlopeState</u>	<p>Turns ON/OFF the power slope. The power slope is valid for the frequency sweep type: Linear, Logarithmic, Segment.</p> <p>Used as attributeID with functions:</p>

	Name	Description
		SetAttributeViBoolean GetAttributeViBoolean The repeated capability identifier: The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.
	StimulusPoints	Sets/Gets the number of sweep points. Used as attributeID with functions: SetAttributeViInt32 GetAttributeViInt32 The repeated capability identifier: The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.
	StimulusPowerCenter	Sets/Gets the center value of the power sweep type. The value is expressed in decibels above 1 milliwatt (dBm). Used as attributeID with functions: SetAttributeViReal64 GetAttributeViReal64 The repeated capability identifier:

	Name	Description
		The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.
	StimulusPowerCwFrequency	<p>Sets/Gets the fixed frequency value when the power sweep type is selected. The value is expressed in hertz (Hz).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	StimulusPowerSpan	<p>Sets/Gets the power span when the power sweep type is active. The value is expressed in decibels above 1 milliwatt (dBm).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>

	Name	Description
	<u>StimulusPowerStart</u>	<p>Sets/Gets the power sweep start value when the power sweep type is active. The value is expressed in decibels above 1 milliwatt (dBm).</p> <p>Used as attributeID with functions:</p> <p><u>SetAttributeViReal64</u></p> <p><u>GetAttributeViReal64</u></p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	<u>StimulusPowerStop</u>	<p>Sets/Gets the power sweep stop value when the power sweep type is active. The value is expressed in decibels above 1 milliwatt (dBm).</p> <p>Used as attributeID with functions:</p> <p><u>SetAttributeViReal64</u></p> <p><u>GetAttributeViReal64</u></p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	<u>StimulusSegmentDisplayOrder</u>	<p>Sets/Gets the display method of the graph horizontal axis for the segment sweep.</p> <p>Used as attributeID with functions:</p>

	Name	Description
		SetAttributeViInt32 GetAttributeViInt32 Field Value SegmentFrequencyOrder SegmentIndexOrder The repeated capability identifier: The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.
	StimulusSweepMeasureDelay	Sets/Gets the delay before measurement in each measurement point. The value is expressed in seconds (sec). Used as attributeID with functions: SetAttributeViReal64 GetAttributeViReal64 The repeated capability identifier: The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.
	StimulusSweepType	Sets/Gets the sweep type of channel. Used as attributeID with functions:

	Name	Description
		SetAttributeViInt32 GetAttributeViInt32 Field Value SweepTypeLinFrequency SweepTypeLogFrequency SweepTypeSegment SweepTypePower The repeated capability identifier: The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.
	StimulusTriggerMode	Sets/Gets the trigger mode. Used as attributeID with functions: SetAttributeViInt32 GetAttributeViInt32 Field Value TriggerModeHold TriggerModeSingle TriggerModeContinuous The repeated capability identifier:



	Name	Description
		The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.
	StimulusTriggerOutput	<p>Turns ON/OFF the trigger output.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	StimulusTriggerOutputFunction	<p>Sets/Gets the trigger output function.</p> <p>The trigger output outputs various waveforms depending on the setting of the Output Trigger Function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>Field Value</p> <p>TriggerOutputFunctionAsampling</p> <p>TriggerOutputFunctionBsampling</p> <p>TriggerOutputFunctionBsetup</p>

	Name	Description
		TriggerOutputFunctionMeasurement TriggerOutputFunctionReadyForTrigger TriggerOutputFunctionSweepEnd The repeated capability identifier: Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.
	StimulusTriggerOutputPolarity	Sets/Gets the polarity of the trigger output. Used as attributeID with functions: SetAttributeViInt32 GetAttributeViInt32 Field Value TriggerOutputPolarityNegative TriggerOutputPolarityPositive The repeated capability identifier: Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.
	StimulusTriggerScope	Sets/Gets the trigger scope. The trigger scope determines the response on the trigger signal arrival: either starts a sweep of all



	Name	Description
		<p>waiting repCapIDs in turn or starts a sweep in the active channel only.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>Field Value</p> <p>TriggerScopeActiveChannel</p> <p>TriggerScopeAllChannel</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	StimulusTriggerSource	<p>Sets/Gets the trigger source.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>Field Value</p> <p>TriggerSourceInternal</p> <p>TriggerSourceExternal</p> <p>TriggerSourceManual</p> <p>TriggerSourceBus</p>



	Name	Description
		<p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	StimulusTriggerState	<p>Gets the the current state of the analyzer.</p> <p>Used as attributeID with functions: GetAttributeViInt32</p> <p>Field Value</p> <p>StimulusTriggerStateHold</p> <p>StimulusTriggerStateMeasure</p> <p>StimulusTriggerStateWait</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	SystemCorrection	<p>Turns ON/OFF the system correction.</p> <p>The system correction is the factory full 1-port calibration performed at the port connectors.</p> <p>Used as attributeID with functions: SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p>



	Name	Description
		Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.
	SystemReadWriteLock	<p>Sets the Analyzer to the local operation mode or remote operation mode, when all the keys on the front panel, mouse and the touch screen are active.</p> <p>Used as attributeID with function SetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	SystemTimeOutMilliseconds	<p>I/O timeout value in milliseconds. This value is expressed in milliseconds (ms).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	ThruAdditionCutoffFreq	<p>Sets/Gets the value of the cutoff frequency of the waveguide thru in the thru addition function.</p> <p>Used as attributeID with functions:</p>



	Name	Description
		SetAttributeViReal64 GetAttributeViReal64 The repeated capability identifier: The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.
	ThruAdditionDelay	Sets/Gets the approximate delay value of an unknown thru in the thru addition function. This value is used to eliminate the uncertainty of ± 180 deg when calculating the phase response of the thru. If this value is set to zero, the analyzer uses an algorithm to automatically determine the delay of the thru. Used as attributeID with functions: SetAttributeViReal64 GetAttributeViReal64 The repeated capability identifier: The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.
	ThruAdditionLength	Sets/Gets the approximate value of the mechanical length of an unknown thru in the thru addition function.



	Name	Description
		<p>This value is used to eliminate the uncertainty of ± 180 deg when calculating the phase response of the thru. If this value is set to zero, the analyzer uses an algorithm to automatically determine the delay of the thru.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	ThruAdditionMedia	<p>Sets/Gets the media of the thru in the thru addition function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>Field Value</p> <p>ThruAdditionMediaCoaxial</p> <p>ThruAdditionMediaWaveguide</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding</p>

	Name	Description
		Channel index i.e. "Channel1" and so on.
	ThruAdditionPermittivity	<p>Sets/Gets the value of the permittivity of the thru media in the thru addition function.</p> <p>This parameter is used to calculate the adapter delay when the thru length is being set; therefore this parameter must be set before setting of the thru length.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	ThruAdditionUnit	<p>Sets/Gets the display units of the thru delay (length) in the thru addition function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>Field Value</p> <p>ThruAdditionUnitMeters</p>

	Name	Description
		<p>ThruAdditionUnitSeconds</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	TimeDomainCableCorrection	<p>Turns ON/OFF the cable correction when the time domain transformation function is turned ON.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	TimeDomainCableFrequency	<p>Sets/Gets the frequency value at which the cable loss is specified for the cable correction function, when the time domain transformation function is turned ON. The value is expressed in Hertz (Hz).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p>


	Name	Description
		<p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	TimeDomainCableLoss	<p>Sets/Gets the cable loss value for the cable correction function, when the time domain transformation function is turned ON.</p> <p>The value is expressed in decibell/meter (dB/m).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	TimeDomainCableVelocityFactor	<p>Sets/Gets the cable relative wave speed velocity for the cable correction function, when the time domain transformation function is turned ON.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p>


	Name	Description
		<p>The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.</p>
	TimeDomainCenter	<p>Sets/Gets the time domain center value, when the time domain transformation function is turned ON. The value is expressed in second (sec).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	TimeDomainGating	<p>Turns ON/OFF the gating function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViBoolean</p> <p>GetAttributeViBoolean</p> <p>The repeated capability identifier:</p>



	Name	Description
		<p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	TimeDomainGatingCenter	<p>Sets/Gets the gate center value of the gating function. The value is expressed in seconds (sec).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	TimeDomainGatingShape	<p>Sets/Gets the gate shape of the gating function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p>

	Name	Description
		<p>GetAttributeViInt32</p> <p>Field Value</p> <p>TimeDomainGatingShapeMaximum</p> <p>TimeDomainGatingShapeMinimum</p> <p>TimeDomainGatingShapeNormal</p> <p>TimeDomainGatingShapeWide</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	TimeDomainGatingSpan	<p>Sets/Gets the gate span value of the gating function. The value is expressed in seconds (sec).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p>


	Name	Description
		<p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	TimeDomainGatingStart	<p>Sets/Gets the gate start value of the gating function. The value is expressed in seconds (sec).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	TimeDomainGatingStop	<p>Sets/Gets the gate stop value of the gating function. The value is expressed in seconds (sec).</p> <p>Used as attributeID with functions:</p>


	Name	Description
		<p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	TimeDomainGatingType	<p>Sets/Gets the gate type of the gating function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>Field Value</p> <p>TimeDomainGatingBandpass</p> <p>TimeDomainGatingNotch</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the</p>

	Name	Description
		corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.
	TimeDomainImpulseWidth	<p>Sets/Gets the impulse width (time domain transformation resolution), coupled with the Kaiser–Bessel window shape Beta parameter.</p> <p>The impulse width setting changes the Beta parameter, and setting of Beta parameter changes the impulse width.</p> <p>The value is expressed in seconds (sec).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	TimeDomainKaiserBeta	Sets/Gets the Beta parameter, which controls the Kaiser–Bessel


	Name	Description
		<p>window shape, when performing time domain transformation.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p>
	TimeDomainSpan	<p>Sets/Gets the time domain span value, when the time domain transformation function is turned ON. The value is expressed in second (sec).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	TimeDomainStart	<p>Sets/Gets the time domain start value, when the time domain transformation function is turned</p>

	Name	Description
		<p>ON. The value is expressed in second (sec).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.</p>
	TimeDomainStop	<p>Sets/Gets the time domain stop value, when the time domain transformation function is turned ON.</p> <p>The value is expressed in second (sec).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViReal64</p> <p>GetAttributeViReal64</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding</p>

	Name	Description
		Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.
	TimeDomainTransformType	<p>Sets/Gets the stimulus type for the time domain transformation function.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeVInt32</p> <p>GetAttributeVInt32</p> <p>Field Value</p> <p>TimeDomainTransformTypeBandpass</p> <p>TimeDomainTransformTypeLowpassImpulse</p> <p>TimeDomainTransformTypeLowpassStep</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e.</p>

	Name	Description
		"Channel1:Measurement3" and so on.
	TimeDomainWindowShape	<p>Sets/Gets the Kaiser–Bessel window shape, when performing time domain transformation.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeVInt32</p> <p>GetAttributeVInt32</p> <p>Field Value</p> <p>TimeDomainWindowShapeArbitrarily</p> <p>TimeDomainWindowShapeMaximum</p> <p>TimeDomainWindowShapeMinimum</p> <p>TimeDomainWindowShapeNormal</p> <p>The repeated capability identifier:</p> <p>The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.</p>

	Name	Description
	VerificationInterval	<p>Sets/Gets the interval between Instrument Performance Verifications.</p> <p>One year (365 days) is recommended.</p> <p>This value is expressed in days.</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViInt32</p> <p>GetAttributeViInt32</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>
	VerificationLastDate	<p>Sets/Gets the date of the last Instrument Performance Verification.</p> <p>The date format: "YYYY,MM,DD" (YYYY - year, MM - month, DD - day).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViString</p> <p>GetAttributeViString</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>

	Name	Description
	VerificationNextDate	<p>Gets the date of the next Instrument Performance Verification.</p> <p>The date format: "YYYY,MM,DD" (YYYY - year, MM - month, DD - day).</p> <p>Used as attributeID with functions:</p> <p>SetAttributeViString</p> <p>GetAttributeViString</p> <p>The repeated capability identifier:</p> <p>Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.</p>

AdapterRemovalCutoffFreq

Description

Sets/Gets the value of the cutoff frequency of the waveguide adapter.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AdapterRemovalCutoffFreq As Integer
```

C#

```
public static int AdapterRemovalCutoffFreq
```

Visual C++

```
public:  
static int AdapterRemovalCutoffFreq
```

JavaScript

```
CMT.Instruments.Attributes.adapterRemovalCutoffFreq
```

Back to [Attributes](#)

AdapterRemovalDelay

Description

Sets/Gets the approximate delay value of an adapter in the adapter removal/insertion function.

This value is used to eliminate the uncertainty of ± 180 deg when calculating the phase response of the adapter.

The sign of the value depends on the type of the removal / insertion function.

The value must be negative for the adapter removal function and must be positive for the adapter insertion function.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AdapterRemovalDelay As Integer
```

C#

```
public static int AdapterRemovalDelay
```

Visual C++

```
public:  
static int AdapterRemovalDelay
```


JavaScript

CMT.Instruments.Attributes.adapterRemovalDelay

Back to [Attributes](#)

AdapterRemovalLength

Description

Sets/Gets the approximate value of the mechanical length of the adapter in the adapter removal/insertion function.

This value is used to eliminate the uncertainty of ± 180 deg when calculating the phase response of the adapter.

The sign of the value depends on the type of the removal / insertion function.

The value must be negative for the adapter removal function and must be positive for the adapter insertion function.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AdapterRemovalLength As Integer
```

C#

```
public static int AdapterRemovalLength
```

Visual C++

```
public:  
static int AdapterRemovalLength
```

JavaScript

CMT.Instruments.Attributes.adapterRemovalLength

Back to [Attributes](#)

AdapterRemovalMedia

Description

Sets/Gets the adapter media in the adapter removal/insertion function.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[AdapterRemovalMediaCoaxial](#)

[AdapterRemovalMediaWaveguide](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AdapterRemovalMedia As Integer
```

C#

```
public static int AdapterRemovalMedia
```

Visual C++

```
public:  
static int AdapterRemovalMedia
```

JavaScript

CMT.Instruments.Attributes.adapterRemovalMedia

Back to [Attributes](#)

AdapterRemovalPermittivity

Description

Sets/Gets the value of the permittivity of an adapter media in the adapter removal/insertion function.

When setting the adapter length, this parameter is used to calculate the adapter delay; therefore this parameter must be set before setting of the adapter length. When setting the adapter delay, this parameter is not used.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AdapterRemovalPermittivity As Integer
```

C#

```
public static int AdapterRemovalPermittivity
```

Visual C++

```
public:  
static int AdapterRemovalPermittivity
```

JavaScript

```
CMT.Instruments.Attributes.adapterRemovalPermittivity
```

Back to [Attributes](#)

AdapterRemovalPort

Description

Sets the port number and sets the adapter removal/insertion function for the calculation of the calibration coefficients when [ApplyCalibration](#) function has been executed.

Used as attributeID with function [SetAttributeViInt32](#).

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AdapterRemovalPort As Integer
```

C#

```
public static int AdapterRemovalPort
```

Visual C++

```
public:  
static int AdapterRemovalPort
```

JavaScript

```
CMT.Instruments.Attributes.adapterRemovalPort
```

Back to [Attributes](#)

AdapterRemovalUnit

Description

Sets/Gets the display units of the adapter delay (length) in the adapter removal/insertion function.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[AdapterRemovalUnitMeters](#)

[AdapterRemovalUnitSeconds](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AdapterRemovalUnit As Integer
```

C#

```
public static int AdapterRemovalUnit
```

Visual C++

```
public:  
static int AdapterRemovalUnit
```

JavaScript

CMT.Instruments.Attributes.adapterRemovalUnit

Back to [Attributes](#)

AnalysisConversion

Description

Turns ON/OFF the S-parameter conversion function.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisConversion As Integer
```

C#

```
public static int AnalysisConversion
```

Visual C++

```
public:  
static int AnalysisConversion
```

JavaScript

```
CMT.Instruments.Attributes.analysisConversion
```

Back to [Attributes](#)

AnalysisConversionFunction

Description

Sets/Gets the S-parameter conversion function type.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[ConversionFunctionConjugation](#)

[ConversionFunctionSInverse](#)

[ConversionFunctionYReflection](#)

[ConversionFunctionYTransmission](#)

[ConversionFunctionYTransShunt](#)

[ConversionFunctionZReflection](#)

[ConversionFunctionZTransmission](#)

[ConversionFunctionZTransShunt](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisConversionFunction As Integer
```

C#

```
public static int AnalysisConversionFunction
```

Visual C++

```
public:  
static int AnalysisConversionFunction
```

JavaScript

```
CMT.Instruments.Attributes.analysisConversionFunction
```

Back to [Attributes](#)

AnalysisDeembedding

Description

Turns ON/OFF the 2-port network de-embedding function.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisDeembedding As Integer
```

C#

```
public static int AnalysisDeembedding
```

Visual C++

```
public:  
static int AnalysisDeembedding
```

JavaScript

```
CMT.Instruments.Attributes.analysisDeembedding
```

Back to [Attributes](#)

AnalysisDeembeddingPort

Description

Turns ON/OFF the 2-port network de-embedding function for specified port.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisDeembeddingPort As Integer
```

C#

```
public static int AnalysisDeembeddingPort
```

Visual C++

```
public:  
static int AnalysisDeembeddingPort
```

JavaScript

```
CMT.Instruments.Attributes.analysisDeembeddingPort
```

Back to [Attributes](#)

AnalysisDeembeddingPortFile

Description

Sets/Gets the name of *.s2p file of the de-embedded circuit of the 2-port network de-embedding function.

The file contains the circuit S-parameters in Touchstone format.

Used as attributeID with functions:

[SetAttributeViString](#)

[GetAttributeViString](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisDeembeddingPortFile As Integer
```

C#

```
public static int AnalysisDeembeddingPortFile
```

Visual C++

```
public:  
static int AnalysisDeembeddingPortFile
```

JavaScript

```
CMT.Instruments.Attributes.analysisDeembeddingPortFile
```

Back to [Attributes](#)

AnalysisEmbedding

Description

Turns ON/OFF the 2-port network embedding function.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisEmbedding As Integer
```

C#

```
public static int AnalysisEmbedding
```

Visual C++

```
public:  
static int AnalysisEmbedding
```

JavaScript

```
CMT.Instruments.Attributes.analysisEmbedding
```

Back to [Attributes](#)

AnalysisEmbeddingPort

Description

Turns ON/OFF the 2-port network embedding function for each port.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisEmbeddingPort As Integer
```

C#

```
public static int AnalysisEmbeddingPort
```

Visual C++

```
public:  
static int AnalysisEmbeddingPort
```

JavaScript

```
CMT.Instruments.Attributes.analysisEmbeddingPort
```

Back to [Attributes](#)

AnalysisEmbeddingPortFile

Description

Sets/Gets the name of *.s2p file of the embedded circuit of the 2-port network embedding function.

The file contains the circuit S-parameters in Touchstone format.

Used as attributeID with functions:

[SetAttributeViString](#)

[GetAttributeViString](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisEmbeddingPortFile As Integer
```

C#

```
public static int AnalysisEmbeddingPortFile
```

Visual C++

```
public:  
static int AnalysisEmbeddingPortFile
```

JavaScript

```
CMT.Instruments.Attributes.analysisEmbeddingPortFile
```

Back to [Attributes](#)

AnalysisFixtureSimulator

Description

Turns ON/OFF the fixture simulation function.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisFixtureSimulator As Integer
```

C#

```
public static int AnalysisFixtureSimulator
```

Visual C++

```
public:  
static int AnalysisFixtureSimulator
```

JavaScript

```
CMT.Instruments.Attributes.analysisFixtureSimulator
```

Back to [Attributes](#)

AnalysisLimitLineDisplay

Description

Turns ON/OFF the limit line display of the limit test function.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisLimitLineDisplay As Integer
```

C#

```
public static int AnalysisLimitLineDisplay
```

Visual C++

```
public:  
static int AnalysisLimitLineDisplay
```

JavaScript

```
CMT.Instruments.Attributes.analysisLimitLineDisplay
```

Back to [Attributes](#)

AnalysisLimitTest

Description

Turns ON/OFF the limit test.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisLimitTest As Integer
```

C#

```
public static int AnalysisLimitTest
```

Visual C++

```
public:  
static int AnalysisLimitTest
```

JavaScript

```
CMT.Instruments.Attributes.analysisLimitTest
```

Back to [Attributes](#)

AnalysisLimitTestFailSign

Description

Turns ON/OFF the "Fail" sign display, when performing limit test.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisLimitTestFailSign As Integer
```

C#

```
public static int AnalysisLimitTestFailSign
```

Visual C++

```
public:  
static int AnalysisLimitTestFailSign
```

JavaScript

```
CMT.Instruments.Attributes.analysisLimitTestFailSign
```

Back to [Attributes](#)

AnalysisLimitTestPoints

Description

Gets the number of the measurement points that failed the limit test.

The stimulus data array of these points can be read out by [GetLimitTestReport](#) function.

Used as attributeID with function [GetAttributeViInt32](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisLimitTestPoints As Integer
```

C#

```
public static int AnalysisLimitTestPoints
```

Visual C++

```
public:  
static int AnalysisLimitTestPoints
```

JavaScript

```
CMT.Instruments.Attributes.analysisLimitTestPoints
```

Back to [Attributes](#)

AnalysisPortZConversion

Description

Turns ON/OFF the port impedance conversion function.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisPortZConversion As Integer
```

C#

```
public static int AnalysisPortZConversion
```

Visual C++

```
public:  
static int AnalysisPortZConversion
```

JavaScript

```
CMT.Instruments.Attributes.analysisPortZConversion
```

Back to [Attributes](#)

AnalysisPortZConversionImag

Description

Sets/ Gets the imaginary part of the impedance of the port impedance conversion function. The value is expressed in Ohm.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisPortZConversionImag As Integer
```

C#

```
public static int AnalysisPortZConversionImag
```

Visual C++

```
public:  
static int AnalysisPortZConversionImag
```

JavaScript

```
CMT.Instruments.Attributes.analysisPortZConversionImag
```

Back to [Attributes](#)

AnalysisPortZConversionReal

Description

Sets/ Gets the real part of the impedance of the port impedance conversion function. The value is expressed in Ohm.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisPortZConversionReal As Integer
```

C#

```
public static int AnalysisPortZConversionReal
```

Visual C++

```
public:  
static int AnalysisPortZConversionReal
```

JavaScript

```
CMT.Instruments.Attributes.analysisPortZConversionReal
```

Back to [Attributes](#)

AnalysisPortZConversionZ0

Description

Sets/Gets the value of the impedance for port impedance conversion function. The value is expressed in Ohm.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisPortZConversionZ0 As Integer
```

C#

```
public static int AnalysisPortZConversionZ0
```

Visual C++

```
public:  
static int AnalysisPortZConversionZ0
```

JavaScript

```
CMT.Instruments.Attributes.analysisPortZConversionZ0
```

Back to [Attributes](#)

AnalysisRippleLimitDisplay

Description

Turns ON/OFF the ripple limit line display.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisRippleLimitDisplay As Integer
```

C#

```
public static int AnalysisRippleLimitDisplay
```

Visual C++

```
public:  
static int AnalysisRippleLimitDisplay
```

JavaScript

```
CMT.Instruments.Attributes.analysisRippleLimitDisplay
```

Back to [Attributes](#)

AnalysisRippleLimitFailSign

Description

Turns ON/OFF the "Fail" sign display for ripple limit test.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisRippleLimitFailSign As Integer
```

C#

```
public static int AnalysisRippleLimitFailSign
```

Visual C++

```
public:  
static int AnalysisRippleLimitFailSign
```

JavaScript

```
CMT.Instruments.Attributes.analysisRippleLimitFailSign
```

Back to [Attributes](#)

AnalysisRippleTest

Description

Turns ON/OFF the ripple limit test.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisRippleTest As Integer
```

C#

```
public static int AnalysisRippleTest
```

Visual C++

```
public:  
static int AnalysisRippleTest
```

JavaScript

```
CMT.Instruments.Attributes.analysisRippleTest
```

Back to [Attributes](#)

AnalysisRippleValueType

Description

Sets/Gets the display type of the ripple value in the specified band.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[AnalysisRippleValueAbsolute](#)

[AnalysisRippleValueMargin](#)

[AnalysisRippleValueOff](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisRippleValueType As Integer
```

C#

```
public static int AnalysisRippleValueType
```

Visual C++

```
public:  
static int AnalysisRippleValueType
```


Back to [Attributes](#)

AnalysisTimeDomain

Description

Turns ON/OFF the time domain transformation function.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisTimeDomain As Integer
```

C#

```
public static int AnalysisTimeDomain
```

Visual C++

```
public:  
static int AnalysisTimeDomain
```

JavaScript

```
CMT.Instruments.Attributes.analysisTimeDomain
```

Back to [Attributes](#)

AnalysisTimeDomainReflectionType

Description

Sets/Gets the reflection distance either one way or round trip for the time domain transformation function.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[TimeDomainReflectionOneWay](#)

[TimeDomainReflectionRoundTrip](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisTimeDomainReflectionType As Integer
```

C#

```
public static int AnalysisTimeDomainReflectionType
```

Visual C++

```
public:  
static int AnalysisTimeDomainReflectionType
```

Back to [Attributes](#)

AnalysisTimeDomainUnits

Description

Sets/Gets the transformation unit for the time domain transformation function: seconds, meters, feet.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[TimeDomainUnitsFeet](#)

[TimeDomainUnitsMeters](#)

[TimeDomainUnitsSeconds](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisTimeDomainUnits As Integer
```

C#

```
public static int AnalysisTimeDomainUnits
```

Visual C++

```
public:
```

Visual C++

```
static int AnalysisTimeDomainUnits
```

JavaScript

```
CMT.Instruments.Attributes.analysisTimeDomainUnits
```

Back to [Attributes](#)

AttrPrintColor

Description

Sets/Gets the color chart for the image printout.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[PrintBlackAndWhite](#)

[PrintColor](#)

[PrintGrayScale](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AttrPrintColor As Integer
```

C#

```
public static int AttrPrintColor
```

Visual C++

```
public:  
static int AttrPrintColor
```

JavaScript

CMT.Instruments.Attributes.attrPrintColor

Back to [Attributes](#)

AutocalAutoOrientation

Description

Turns ON/OFF the Auto-Orientation function when the AutoCal Module calibration is executed.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AutocalAutoOrientation As Integer
```

C#

```
public static int AutocalAutoOrientation
```

Visual C++

```
public:  
static int AutocalAutoOrientation
```

JavaScript

```
CMT.Instruments.Attributes.autocalAutoOrientation
```

Back to [Attributes](#)

AutocalCharacterization

Description

Sets/Gets the characterization number used when executing AutoCal (factory or user characterizations).

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[AutocalCharacterizationFactory](#)

[AutocalCharacterizationUser1](#)

[AutocalCharacterizationUser2](#)

[AutocalCharacterizationUser3](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AutocalCharacterization As Integer
```

C#

```
public static int AutocalCharacterization
```

Visual C++

```
public:
```

Visual C++

```
static int AutocalCharacterization
```

JavaScript

```
CMT.Instruments.Attributes.autocalCharacterization
```

Back to [Attributes](#)

AutocalModuleReady

Description

Gets the ready state of the AutoCal module. The state is True when the AutoCal module is connected.

Used as attributeID with function [GetAttributeViBoolean](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AutocalModuleReady As Integer
```

C#

```
public static int AutocalModuleReady
```

Visual C++

```
public:  
static int AutocalModuleReady
```

JavaScript

```
CMT.Instruments.Attributes.autocalModuleReady
```

Back to [Attributes](#)

AutocalOrientationPort

Description

Sets/Gets the AutoCal module port number which is connected to a specified port of Network Analyzer.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[AutocalOrientationPortA](#)

[AutocalOrientationPortB](#)

[AutocalOrientationPortC](#)

[AutocalOrientationPortD](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AutocalOrientationPort As Integer
```

C#

```
public static int AutocalOrientationPort
```

Visual C++

```
public:
```

Visual C++

```
static int AutocalOrientationPort
```

JavaScript

```
CMT.Instruments.Attributes.autocalOrientationPort
```

Back to [Attributes](#)

AutocalTemperature

Description

Gets the temperature of the AutoCal module connected to the Analyzer. The value is expressed in Celsius degrees.

Used as attributeID with functions: [GetAttributeViReal64](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AutocalTemperature As Integer
```

C#

```
public static int AutocalTemperature
```

Visual C++

```
public:  
static int AutocalTemperature
```

JavaScript

```
CMT.Instruments.Attributes.autocalTemperature
```

Back to [Attributes](#)

AutocalUnknownThru

Description

Turns ON/OFF the Unknown Thru feature when the AutoCal Module calibration is executed.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AutocalUnknownThru As Integer
```

C#

```
public static int AutocalUnknownThru
```

Visual C++

```
public:  
static int AutocalUnknownThru
```

JavaScript

```
CMT.Instruments.Attributes.autocalUnknownThru
```

Back to [Attributes](#)

AutoPortExtensionAdjustMismatch

Description

Turns ON/OFF the usage of "Loss at DC" value for the results of the auto port extension function.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AutoPortExtensionAdjustMismatch As Integer
```

C#

```
public static int AutoPortExtensionAdjustMismatch
```

Visual C++

```
public:  
static int AutoPortExtensionAdjustMismatch
```

JavaScript

```
CMT.Instruments.Attributes.autoPortExtensionAdjustMismatch
```

Back to [Attributes](#)

AutoPortExtensionIncludeLoss

Description

Turns ON/OFF the usage of "Loss1" and "Loss2" values for the results of the auto port extension function.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AutoPortExtensionIncludeLoss As Integer
```

C#

```
public static int AutoPortExtensionIncludeLoss
```

Visual C++

```
public:  
static int AutoPortExtensionIncludeLoss
```

JavaScript

```
CMT.Instruments.Attributes.autoPortExtensionIncludeLoss
```

Back to [Attributes](#)

AutoPortExtensionMethod

Description

Sets/Gets the frequency range used for calculation of the results of the Auto Port Extension function.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[AutoPortExtensionActiveMarker](#)

[AutoPortExtensionCurrentSpan](#)

[AutoPortExtensionUserSpan](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AutoPortExtensionMethod As Integer
```

C#

```
public static int AutoPortExtensionMethod
```

Visual C++

```
public:  
static int AutoPortExtensionMethod
```

JavaScript

CMT.Instruments.Attributes.autoPortExtensionMethod

Back to [Attributes](#)

AutoPortExtensionUserSpanStart

Description

Sets/Gets the start value of the user span of the auto port extension function. The value is expressed in Hertz (Hz).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AutoPortExtensionUserSpanStart As Integer
```

C#

```
public static int AutoPortExtensionUserSpanStart
```

Visual C++

```
public:  
static int AutoPortExtensionUserSpanStart
```

JavaScript

```
CMT.Instruments.Attributes.autoPortExtensionUserSpanStart
```

Back to [Attributes](#)

AutoPortExtensionUserSpanStop

Description

Sets/Gets the stop value of the user span of the auto port extension function. The value is expressed in Hertz (Hz).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AutoPortExtensionUserSpanStop As Integer
```

C#

```
public static int AutoPortExtensionUserSpanStop
```

Visual C++

```
public:  
static int AutoPortExtensionUserSpanStop
```

JavaScript

```
CMT.Instruments.Attributes.autoPortExtensionUserSpanStop
```

Back to [Attributes](#)

BeepCompleteOn

Description

Turns ON/OFF the beeper notifying of the completion of the operation.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared BeepCompleteOn As Integer
```

C#

```
public static int BeepCompleteOn
```

Visual C++

```
public:  
static int BeepCompleteOn
```

JavaScript

```
CMT.Instruments.Attributes.beepCompleteOn
```

Back to [Attributes](#)

BeepWarning

Description

Turns ON/OFF the beeper notifying of warning.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared BeepWarning As Integer
```

C#

```
public static int BeepWarning
```

Visual C++

```
public:  
static int BeepWarning
```

JavaScript

```
CMT.Instruments.Attributes.beepWarning
```

Back to [Attributes](#)

CalibrationAutoSelectZ0

Description

Turns ON/OFF the auto-select Z0 function.

When enabled the function sets the port impedance Z0 to the corresponding value of measuring calibration standard.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalibrationAutoSelectZ0 As Integer
```

C#

```
public static int CalibrationAutoSelectZ0
```

Visual C++

```
public:  
static int CalibrationAutoSelectZ0
```

JavaScript

```
CMT.Instruments.Attributes.calibrationAutoSelectZ0
```

Back to [Attributes](#)

CalibrationCorrection

Description

Turns ON/OFF the S-parameter error correction.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalibrationCorrection As Integer
```

C#

```
public static int CalibrationCorrection
```

Visual C++

```
public:  
static int CalibrationCorrection
```

JavaScript

```
CMT.Instruments.Attributes.calibrationCorrection
```

Back to [Attributes](#)

CalibrationPortZ0

Description

Gets the system impedance Z0 or the the impedance Z0 of port. The value is expressed in Ohm.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

Must be VI_NULL or an empty string if is used the system impedance Z0. If id used the the impedance Z0 of port, the physical names are supported along with the corresponding Port index i.e. "Port1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalibrationPortZ0 As Integer
```

C#

```
public static int CalibrationPortZ0
```

Visual C++

```
public:  
static int CalibrationPortZ0
```

JavaScript

```
CMT.Instruments.Attributes.calibrationPortZ0
```

Back to [Attributes](#)

CalibrationTriggerSource

Description

Enables/Disables the internal trigger source for calibration.

If the internal trigger source for calibration is enabled then a command of the calibration standard measurement starts the measurement immediately. If the internal trigger source for calibration is disabled then the system trigger source is used (which is set for regular measurement with [StimulusTriggerSource](#) to start the calibration standard measurement. The system trigger source also enables the averaging trigger [MeasurementAvgTrigger](#) and the point trigger [StimulusExtTriggerEvent](#) for calibration.

NOTE

When the system trigger source is selected you should avoid the program trigger source (BUS), otherwise the program deadlock is possible.

NOTE

The command isn't applied to the electronic calibration, the power calibration and the receiver calibration. The internal trigger is always used in these cases.

Field Value

[CalibrationTriggerSourceInternal](#)

[CalibrationTriggerSourceSystem](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalibrationTriggerSource As Integer
```

C#

```
public static int CalibrationTriggerSource
```

Visual C++

```
public:  
static int CalibrationTriggerSource
```

JavaScript

```
CMT.Instruments.Attributes.calibrationTriggerSource
```

Back to [Attributes](#)

CalibrationType

Description

Gets the calibration type for the calculation of the calibration coefficients on completion of the calibration executed by [ApplyCalibration](#) function.

Used as attributeID with function [GetAttributeViInt32](#)

Field Value

[CalibrationType1path2port](#)

[CalibrationType1portSol](#)

[CalibrationType2portSolt](#)

[CalibrationType2portTrl](#)

[CalibrationTypeNotDefined](#)

[CalibrationTypeResponseOpen](#)

[CalibrationTypeResponseShort](#)

[CalibrationTypeResponseThru](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalibrationType As Integer
```

C#

```
public static int CalibrationType
```

Visual C++

```
public:  
static int CalibrationType
```

JavaScript

```
CMT.Instruments.Attributes.calibrationType
```

Back to [Attributes](#)

CalkitDescription

Description

Sets/Gets the calibration kit description string.

Used as attributeID with functions:

[SetAttributeViString](#)

[GetAttributeViString](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalkitDescription As Integer
```

C#

```
public static int CalkitDescription
```

Visual C++

```
public:  
static int CalkitDescription
```

JavaScript

```
CMT.Instruments.Attributes.calkitDescription
```

Back to [Attributes](#)

CalkitLabel

Description

Sets/Gets the calibration kit label.

Used as attributeID with functions:

[SetAttributeViString](#)

[GetAttributeViString](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalkitLabel As Integer
```

C#

```
public static int CalkitLabel
```

Visual C++

```
public:  
static int CalkitLabel
```

JavaScript

```
CMT.Instruments.Attributes.calkitLabel
```

Back to [Attributes](#)

CalkitSelected

Description

Sets/Gets the number of the selected calibration kit in the table of calibration kits.

The selected calibration kit is used in the subsequent calibration and is used for editing by [CalkitStandardOpen](#), [CalkitStandardShort](#) e.c.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalkitSelected As Integer
```

C#

```
public static int CalkitSelected
```

Visual C++

```
public:  
static int CalkitSelected
```

JavaScript

```
CMT.Instruments.Attributes.calkitSelected
```

Back to [Attributes](#)

CalkitStandardArbitrary

Description

Sets/Gets the value of the arbitrary impedance for the load standard. The value is expressed in Ohm.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalkitStandardArbitrary As Integer
```

C#

```
public static int CalkitStandardArbitrary
```

Visual C++

```
public:  
static int CalkitStandardArbitrary
```

JavaScript

```
CMT.Instruments.Attributes.calkitStandardArbitrary
```

Back to [Attributes](#)

CalkitStandardC0

Description

Sets/Gets the C0 value for the open calibration standard. The value is expressed in 1E-15 F.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalkitStandardC0 As Integer
```

C#

```
public static int CalkitStandardC0
```

Visual C++

```
public:  
static int CalkitStandardC0
```

JavaScript

```
CMT.Instruments.Attributes.calkitStandardC0
```

Back to [Attributes](#)

CalkitStandardC1

Description

Sets/Gets the C1 value for the open calibration standard. The value is expressed in $1\text{E}-27$ F/Hz.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalkitStandardC1 As Integer
```

C#

```
public static int CalkitStandardC1
```

Visual C++

```
public:  
static int CalkitStandardC1
```

JavaScript

```
CMT.Instruments.Attributes.calkitStandardC1
```

Back to [Attributes](#)

CalkitStandardC2

Description

Sets/Gets the C2 value for the open calibration standard. The value is expressed in $1\text{E}-36 \text{ F/Hz}^2$.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalkitStandardC2 As Integer
```

C#

```
public static int CalkitStandardC2
```

Visual C++

```
public:  
static int CalkitStandardC2
```

JavaScript

```
CMT.Instruments.Attributes.calkitStandardC2
```

Back to [Attributes](#)

CalkitStandardC3

Description

Sets/Gets the C3 value for the open calibration standard. The value is expressed in $1\text{E}-45 \text{ F/Hz}^3$.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalkitStandardC3 As Integer
```

C#

```
public static int CalkitStandardC3
```

Visual C++

```
public:  
static int CalkitStandardC3
```

JavaScript

```
CMT.Instruments.Attributes.calkitStandardC3
```

Back to [Attributes](#)

CalkitStandardL0

Description

Sets/Gets the L0 value for the short calibration standard. The value is expressed in $1\text{E}-12$ H.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalkitStandardL0 As Integer
```

C#

```
public static int CalkitStandardL0
```

Visual C++

```
public:  
static int CalkitStandardL0
```

JavaScript

```
CMT.Instruments.Attributes.calkitStandardL0
```

Back to [Attributes](#)

CalkitStandardL1

Description

Sets/Gets the L1 value for the short calibration standard. The value is expressed in 1E-24 H/Hz.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalkitStandardL1 As Integer
```

C#

```
public static int CalkitStandardL1
```

Visual C++

```
public:  
static int CalkitStandardL1
```

JavaScript

```
CMT.Instruments.Attributes.calkitStandardL1
```

Back to [Attributes](#)

CalkitStandardL2

Description

Sets/Gets the L2 value for the short calibration standard. The value is expressed in $1E-33 \text{ H/Hz}^2$.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalkitStandardL2 As Integer
```

C#

```
public static int CalkitStandardL2
```

Visual C++

```
public:  
static int CalkitStandardL2
```

JavaScript

```
CMT.Instruments.Attributes.calkitStandardL2
```

Back to [Attributes](#)

CalkitStandardL3

Description

Sets/Gets the L3 value for the short calibration standard. The value is expressed in $1E-42 \text{ H/Hz}^3$.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalkitStandardL3 As Integer
```

C#

```
public static int CalkitStandardL3
```

Visual C++

```
public:  
static int CalkitStandardL3
```

JavaScript

```
CMT.Instruments.Attributes.calkitStandardL3
```

Back to [Attributes](#)

CalkitStandardLabel

Description

Sets/Gets the label for the calibration standard.

Used as attributeID with functions:

[SetAttributeViString](#)

[GetAttributeViString](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalkitStandardLabel As Integer
```

C#

```
public static int CalkitStandardLabel
```

Visual C++

```
public:  
static int CalkitStandardLabel
```

JavaScript

```
CMT.Instruments.Attributes.calkitStandardLabel
```

Back to [Attributes](#)

CalkitStandardMaxFrequency

Description

Sets/Gets the maximum frequency limit of the calibration standard. The value is expressed in hertz (Hz).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalkitStandardMaxFrequency As Integer
```

C#

```
public static int CalkitStandardMaxFrequency
```

Visual C++

```
public:  
static int CalkitStandardMaxFrequency
```

JavaScript

```
CMT.Instruments.Attributes.calkitStandardMaxFrequency
```

Back to [Attributes](#)

CalkitStandardMinFrequency

Description

Sets/Gets the minimum frequency limit of the calibration standard. The value is expressed in hertz (Hz).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalkitStandardMinFrequency As Integer
```

C#

```
public static int CalkitStandardMinFrequency
```

Visual C++

```
public:  
static int CalkitStandardMinFrequency
```

JavaScript

```
CMT.Instruments.Attributes.calkitStandardMinFrequency
```

Back to [Attributes](#)

CalkitStandardOffsetDelay

Description

Sets/Gets the offset delay value for the calibration standard. The value is expressed in seconds (sec).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalkitStandardOffsetDelay As Integer
```

C#

```
public static int CalkitStandardOffsetDelay
```

Visual C++

```
public:  
static int CalkitStandardOffsetDelay
```

JavaScript

```
CMT.Instruments.Attributes.calkitStandardOffsetDelay
```

Back to [Attributes](#)

CalkitStandardOffsetLoss

Description

Sets/Gets the offset loss value for the calibration standard. The value is expressed in Ohm/seconds.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalkitStandardOffsetLoss As Integer
```

C#

```
public static int CalkitStandardOffsetLoss
```

Visual C++

```
public:  
static int CalkitStandardOffsetLoss
```

JavaScript

```
CMT.Instruments.Attributes.calkitStandardOffsetLoss
```

Back to [Attributes](#)

CalkitStandardOffsetZ0

Description

Sets/Gets the offset Z0 value for the calibration standard. The value is expressed in Ohm.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalkitStandardOffsetZ0 As Integer
```

C#

```
public static int CalkitStandardOffsetZ0
```

Visual C++

```
public:  
static int CalkitStandardOffsetZ0
```

JavaScript

```
CMT.Instruments.Attributes.calkitStandardOffsetZ0
```

Back to [Attributes](#)

CalkitStandardType

Description

Sets/Gets the type of calibration standard.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[CalkitStandardDataBased](#)

[CalkitStandardLoad](#)

[CalkitStandardNone](#)

[CalkitStandardOpen](#)

[CalkitStandardShort](#)

[CalkitStandardSlidingLoad](#)

[CalkitStandardThruDelay](#)

[CalkitStandardUnknThru](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Standard index i.e. "Standard1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalkitStandardType As Integer
```

C#

```
public static int CalkitStandardType
```

Visual C++

```
public:  
static int CalkitStandardType
```

JavaScript

```
CMT.Instruments.Attributes.calkitStandardType
```

Back to [Attributes](#)

CorrectionState

Description

Gets the interpolation/extrapolation status of the error correction.

Used as attributeID with function [GetAttributeViString](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CorrectionState As Integer
```

C#

```
public static int CorrectionState
```

Visual C++

```
public:  
static int CorrectionState
```

JavaScript

```
CMT.Instruments.Attributes.correctionState
```

Back to [Attributes](#)

CorrectionStatus

Description

Gets the interpolation/extrapolation status of the error correction.

Used as attributeID with function [GetAttributeViString](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CorrectionStatus As Integer
```

C#

```
public static int CorrectionStatus
```

Visual C++

```
public:  
static int CorrectionStatus
```

JavaScript

```
CMT.Instruments.Attributes.correctionStatus
```

Back to [Attributes](#)

CorrectionType

Description

Gets the applied calibration type and port numbers for the specified trace.

Used as attributeID with function [GetAttributeViString](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index end Measurement index i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CorrectionType As Integer
```

C#

```
public static int CorrectionType
```

Visual C++

```
public:  
static int CorrectionType
```

JavaScript

```
CMT.Instruments.Attributes.correctionType
```

Back to [Attributes](#)

DevicePxiChassis

Description

Gets the identifier for the PXI chassis in which the CMT vector network analyzer is installed.

Used as attributeID with function [GetAttributeViInt32](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DevicePxiChassis As Integer
```

C#

```
public static int DevicePxiChassis
```

Visual C++

```
public:  
static int DevicePxiChassis
```

JavaScript

```
CMT.Instruments.Attributes.devicePxiChassis
```

Back to [Attributes](#)

DevicePxiSlot

Description

Gets the number for the PXI slot in which the CMT vector network analyzer is installed.

Used as attributeID with function [GetAttributeViInt32](#)

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DevicePxiSlot As Integer
```

C#

```
public static int DevicePxiSlot
```

Visual C++

```
public:  
static int DevicePxiSlot
```

JavaScript

```
CMT.Instruments.Attributes.devicePxiSlot
```

Back to [Attributes](#)

DeviceReady

Description

Gets the ready state of the Analyzer.

The state is True when analyzer hardware is connected, powered and the boot process is completed (about 15 sec).

Used as attributeID with function [GetAttributeViBoolean](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DeviceReady As Integer
```

C#

```
public static int DeviceReady
```

Visual C++

```
public:  
static int DeviceReady
```

JavaScript

```
CMT.Instruments.Attributes.deviceReady
```

Back to [Attributes](#)

DeviceSerialNumber

Description

Gets the instrument serial number.

Used as attributeID with function [GetAttributeViString](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DeviceSerialNumber As Integer
```

C#

```
public static int DeviceSerialNumber
```

Visual C++

```
public:  
static int DeviceSerialNumber
```

JavaScript

```
CMT.Instruments.Attributes.deviceSerialNumber
```

Back to [Attributes](#)

DeviceTemperature

Description

Gets the specified sensor temperature inside the Analyzer. The value is expressed in Celsius degrees.

Used as attributeID with function [GetAttributeViInt32](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DeviceTemperature As Integer
```

C#

```
public static int DeviceTemperature
```

Visual C++

```
public:  
static int DeviceTemperature
```

JavaScript

```
CMT.Instruments.Attributes.deviceTemperature
```

Back to [Attributes](#)

DisplayActiveChannel

Description

Sets/Gets the active channel.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

The repeated capability identifier:

Pass VI_NULL or empty string if operation does not apply to a repeated capability. The physical names are supported along with the corresponding Channel index for the active trace number of the channel i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DisplayActiveChannel As Integer
```

C#

```
public static int DisplayActiveChannel
```

Visual C++

```
public:  
static int DisplayActiveChannel
```

JavaScript

```
CMT.Instruments.Attributes.displayActiveChannel
```

Back to [Attributes](#)

DisplayActiveTrace

Description

Sets/Gets the active trace in channel.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace number of the channel i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DisplayActiveTrace As Integer
```

C#

```
public static int DisplayActiveTrace
```

Visual C++

```
public:  
static int DisplayActiveTrace
```

JavaScript

```
CMT.Instruments.Attributes.displayActiveTrace
```

Back to [Attributes](#)

DisplayBackgroundColor

Description

Sets/Gets the background color for trace display.

The color is defined by its mix of red (RR), green (GG) and blue (BB) components each represented by 16-bit hexadecimal number.

Used as attributeID with functions:

[SetAttributeVInt32](#)

[GetAttributeVInt32](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DisplayBackgroundColor As Integer
```

C#

```
public static int DisplayBackgroundColor
```

Visual C++

```
public:  
static int DisplayBackgroundColor
```

JavaScript

```
CMT.Instruments.Attributes.displayBackgroundColor
```

Back to [Attributes](#)

DisplayChannelAllocation

Description

Sets/Gets the layout of the channel windows on the screen.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DisplayChannelAllocation As Integer
```

C#

```
public static int DisplayChannelAllocation
```

Visual C++

```
public:  
static int DisplayChannelAllocation
```

JavaScript

```
CMT.Instruments.Attributes.displayChannelAllocation
```

Back to [Attributes](#)

DisplayCycleTimeValue

Description

Gets the measured cycle time.

The cycle time is the interval between the start of two adjacent sweeps. The cycle time is averaged by an exponential window with a time constant of about 0.5 sec. If the cycle time is changed more than 100 usec in comparison with the averaged time, the averaging starts anew. The value is expressed in seconds (sec).

Used as attributeID with functions: [GetAttributeViReal64](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DisplayCycleTimeValue As Integer
```

C#

```
public static int DisplayCycleTimeValue
```

Visual C++

```
public:  
static int DisplayCycleTimeValue
```

JavaScript

```
CMT.Instruments.Attributes.displayCycleTimeValue
```

Back to [Attributes](#)

DisplayDataTraceColor

Description

Sets/Gets the data trace color.

The color is defined by its mix of red (RR), green (GG) and blue (BB) components each represented by 16-bit hexadecimal number.

Used as attributeID with functions:

[SetAttributeVInt32](#)

[GetAttributeVInt32](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Measurement index i.e. "Measurement1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DisplayDataTraceColor As Integer
```

C#

```
public static int DisplayDataTraceColor
```

Visual C++

```
public:  
static int DisplayDataTraceColor
```

JavaScript

```
CMT.Instruments.Attributes.displayDataTraceColor
```

Back to [Attributes](#)

DisplayGridColor

Description

Sets/Gets the grid and the graticule label color for trace display.

The color is defined by its mix of red (RR), green (GG) and blue (BB) components each represented by 16-bit hexadecimal number.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DisplayGridColor As Integer
```

C#

```
public static int DisplayGridColor
```

Visual C++

```
public:  
static int DisplayGridColor
```

JavaScript

```
CMT.Instruments.Attributes.displayGridColor
```

Back to [Attributes](#)

DisplayInvertColor

Description

Turns ON/OFF the inversion of display colors of the traces area.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DisplayInvertColor As Integer
```

C#

```
public static int DisplayInvertColor
```

Visual C++

```
public:  
static int DisplayInvertColor
```

JavaScript

```
CMT.Instruments.Attributes.displayInvertColor
```

Back to [Attributes](#)

DisplayMaximizeChannel

Description

Turns ON/OFF of the maximization of the active channel window.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DisplayMaximizeChannel As Integer
```

C#

```
public static int DisplayMaximizeChannel
```

Visual C++

```
public:  
static int DisplayMaximizeChannel
```

JavaScript

```
CMT.Instruments.Attributes.displayMaximizeChannel
```

Back to [Attributes](#)

DisplayMaximizeTrace

Description

Turns ON/OFF the active trace maximization inside the specified channel.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DisplayMaximizeTrace As Integer
```

C#

```
public static int DisplayMaximizeTrace
```

Visual C++

```
public:  
static int DisplayMaximizeTrace
```

JavaScript

```
CMT.Instruments.Attributes.displayMaximizeTrace
```

Back to [Attributes](#)

DisplayMemoryTraceColor

Description

Sets/Gets the memory trace color.

The color is defined by its mix of red (RR), green (GG) and blue (BB) components each represented by 16-bit hexadecimal number.

Used as attributeID with functions:

[SetAttributeVInt32](#)

[GetAttributeVInt32](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Measurement index i.e. "Measurement1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DisplayMemoryTraceColor As Integer
```

C#

```
public static int DisplayMemoryTraceColor
```

Visual C++

```
public:  
static int DisplayMemoryTraceColor
```

JavaScript

```
CMT.Instruments.Attributes.displayMemoryTraceColor
```

Back to [Attributes](#)

DisplaySystemDate

Description

Sets/Gets the current date.

The date format: "YYYY,MM,DD" (YYYY - year, MM - month, DD - day).

Used as attributeID with functions:

[SetAttributeViString](#)

[GetAttributeViString](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DisplaySystemDate As Integer
```

C#

```
public static int DisplaySystemDate
```

Visual C++

```
public:  
static int DisplaySystemDate
```

JavaScript

```
CMT.Instruments.Attributes.displaySystemDate
```

Back to [Attributes](#)

DisplaySystemTime

Description

Sets/Gets the current time.

The time format: "hh,mm,ss" (hh - hour, mm - minute, ss - second).

Used as attributeID with functions:

[SetAttributeViString](#)

[GetAttributeViString](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DisplaySystemTime As Integer
```

C#

```
public static int DisplaySystemTime
```

Visual C++

```
public:  
static int DisplaySystemTime
```

JavaScript

```
CMT.Instruments.Attributes.displaySystemTime
```

Back to [Attributes](#)

DisplayTitleData

Description

Sets/Gets the channel title label.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DisplayTitleData As Integer
```

C#

```
public static int DisplayTitleData
```

Visual C++

```
public:  
static int DisplayTitleData
```

JavaScript

```
CMT.Instruments.Attributes.displayTitleData
```

Back to [Attributes](#)

DisplayTitleLabel

Description

Turns ON/OFF the channel title display.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DisplayTitleLabel As Integer
```

C#

```
public static int DisplayTitleLabel
```

Visual C++

```
public:  
static int DisplayTitleLabel
```

JavaScript

```
CMT.Instruments.Attributes.displayTitleLabel
```

Back to [Attributes](#)

DisplayTraceAllocation

Description

Sets/Gets the layout of the graph in the channel window.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DisplayTraceAllocation As Integer
```

C#

```
public static int DisplayTraceAllocation
```

Visual C++

```
public:  
static int DisplayTraceAllocation
```

JavaScript

```
CMT.Instruments.Attributes.displayTraceAllocation
```

Back to [Attributes](#)

DisplayTraceCount

Description

Sets/Gets the number of traces in the channel.

Used as attributeID with functions:

[SetAttributeVInt32](#)

[GetAttributeVInt32](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DisplayTraceCount As Integer
```

C#

```
public static int DisplayTraceCount
```

Visual C++

```
public:  
static int DisplayTraceCount
```

JavaScript

```
CMT.Instruments.Attributes.displayTraceCount
```

Back to [Attributes](#)

DisplayTraceDataMath

Description

Sets/Gets the math operation between the data trace and the memory trace.

The math result replaces the data trace. If the memory trace does not exist, the command is ignored.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[TraceDataMathAdd](#)

[TraceDataMathDiv](#)

[TraceDataMathMult](#)

[TraceDataMathOff](#)

[TraceDataMathSubt](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DisplayTraceDataMath As Integer
```

C#

```
public static int DisplayTraceDataMath
```

Visual C++

```
public:  
static int DisplayTraceDataMath
```

JavaScript

```
CMT.Instruments.Attributes.displayTraceDataMath
```

Back to [Attributes](#)

DisplayTraceHoldType

Description

Sets/Gets the type of the trace hold function.

The function holds the trace at the maximum or minimum point.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[TraceHoldMax](#)

[TraceHoldMin](#)

[TraceHoldOff](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DisplayTraceHoldType As Integer
```

C#

```
public static int DisplayTraceHoldType
```

Visual C++

```
public:
```

Visual C++

```
static int DisplayTraceHoldType
```

JavaScript

```
CMT.Instruments.Attributes.displayTraceHoldType
```

Back to [Attributes](#)

DisplayTraceType

Description

Turns ON/OFF the memory trace display.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[DisplayTraceData](#)

[DisplayTraceDataAndMemory](#)

[DisplayTraceMemory](#)

[DisplayTraceOff](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Measurement index i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DisplayTraceType As Integer
```

C#

```
public static int DisplayTraceType
```

Visual C++

```
public:  
static int DisplayTraceType
```

JavaScript

CMT.Instruments.Attributes.displayTraceType

Back to [Attributes](#)

DisplayUpdate

Description

Turns ON/OFF the display update.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DisplayUpdate As Integer
```

C#

```
public static int DisplayUpdate
```

Visual C++

```
public:  
static int DisplayUpdate
```

JavaScript

```
CMT.Instruments.Attributes.displayUpdate
```

Back to [Attributes](#)

ExternalReferenceRoute

NOTE

PXle-S5090 model only

Description

Sets/Gets the route of the external 10 MHz reference frequency.

The source of the reference [ReferenceFrequencySource](#) must be set to the EXTernal.

Used as attributeID with functions:

[SetAttributeVInt32](#)

[GetAttributeVInt32](#)

Field Value

[ReferenceRouteFront](#)

[ReferenceRouteRear](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ExternalReferenceRoute As Integer
```

C#

```
public static int ExternalReferenceRoute
```

Visual C++

```
public:  
static int ExternalReferenceRoute
```

Javascript

```
CMT.Instruments.Attributes.externalReferenceRoute
```

Back to [Attributes](#)

ExternalTriggerRoute

NOTE

Command valid for PXIe-S5090 model only.

Description

Sets/Gets the connector to use for the external trigger input in a PXI system.

The trigger source must be set to the EXTernal. One of the 10 routes can be selected.

The same line cannot be selected as input and output trigger route.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[TriggerRouteSmb](#)

[TriggerRoutePxiTrig0](#)

[TriggerRoutePxiTrig1](#)

[TriggerRoutePxiTrig2](#)

[TriggerRoutePxiTrig3](#)

[TriggerRoutePxiTrig4](#)

[TriggerRoutePxiTrig5](#)

[TriggerRoutePxiTrig6](#)

[TriggerRoutePxiTrig7](#)

[TriggerRouteStar](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ExternalTriggerRoute As Integer
```

C#

```
public static int ExternalTriggerRoute
```

Visual C++

```
public:  
static int ExternalTriggerRoute
```

JavaScript

```
CMT.Instruments.Attributes.externalTriggerRoute
```

Back to [Attributes](#)

FrequencyDivider

Description

Sets/Gets the basic frequency range divisor of current port when the frequency offset feature is ON and offset type is "PORT".

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FrequencyDivider As Integer
```

C#

```
public static int FrequencyDivider
```

Visual C++

```
public:  
static int FrequencyDivider
```

JavaScript

```
CMT.Instruments.Attributes.frequencyDivider
```

Back to [Attributes](#)

FrequencyMultiplier

Description

Sets/Gets the basic frequency range multiplier of port when the frequency offset feature is ON and offset type is "PORT".

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FrequencyMultiplier As Integer
```

C#

```
public static int FrequencyMultiplier
```

Visual C++

```
public:  
static int FrequencyMultiplier
```

JavaScript

```
CMT.Instruments.Attributes.frequencyMultiplier
```

Back to [Attributes](#)

FrequencyOffset

Description

Sets/Gets the basic frequency range offset of current port when the frequency offset feature is ON and offset type is "PORT".

The value is expressed in hertz (Hz).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FrequencyOffset As Integer
```

C#

```
public static int FrequencyOffset
```

Visual C++

```
public:  
static int FrequencyOffset
```

JavaScript

```
CMT.Instruments.Attributes.frequencyOffset
```

Back to [Attributes](#)

FrequencyOffsetStart

Description

Sets/Gets the frequency sweep start of current port when the frequency offset feature is ON and offset type is "PORT".

The value is expressed in hertz (Hz).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FrequencyOffsetStart As Integer
```

C#

```
public static int FrequencyOffsetStart
```

Visual C++

```
public:  
static int FrequencyOffsetStart
```

JavaScript

```
CMT.Instruments.Attributes.frequencyOffsetStart
```

Back to [Attributes](#)

FrequencyOffsetStop

Description

Sets/Gets the frequency sweep stop of current port when the frequency offset feature is ON and offset type is "PORT".

The value is expressed in hertz (Hz).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FrequencyOffsetStop As Integer
```

C#

```
public static int FrequencyOffsetStop
```

Visual C++

```
public:  
static int FrequencyOffsetStop
```

JavaScript

```
CMT.Instruments.Attributes.frequencyOffsetStop
```

Back to [Attributes](#)

FrequencyOffsetType

Description

Sets/Gets the frequency offset type when the frequency offset feature is ON.

There are two frequency offset types: "Port1/Port2" and "Source/Receivers". First offset type offsets ports against each other. Second offset type offsets source against receivers.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[FrequencyOffsetTypePortPort](#)

[FrequencyOffsetTypeSourceReceiver](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FrequencyOffsetType As Integer
```

C#

```
public static int FrequencyOffsetType
```

Visual C++

```
public:  
static int FrequencyOffsetType
```

JavaScript

CMT.Instruments.Attributes.frequencyOffsetType

Back to [Attributes](#)

FrequencyReceiverDivider

Description

Sets/Gets the basic frequency range divisor to get the receiver frequency when the frequency offset feature is ON and offset type is "SRCRcv".

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FrequencyReceiverDivider As Integer
```

C#

```
public static int FrequencyReceiverDivider
```

Visual C++

```
public:  
static int FrequencyReceiverDivider
```

JavaScript

```
CMT.Instruments.Attributes.frequencyReceiverDivider
```

Back to [Attributes](#)

FrequencyReceiverMultiplier

Description

Sets/Gets the basic frequency range multiplier to get the receiver frequency when the frequency offset feature is ON and offset type is "SRCRcv".

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FrequencyReceiverMultiplier As Integer
```

C#

```
public static int FrequencyReceiverMultiplier
```

Visual C++

```
public:  
static int FrequencyReceiverMultiplier
```

JavaScript

```
CMT.Instruments.Attributes.frequencyReceiverMultiplier
```

Back to [Attributes](#)

FrequencyReceiverOffset

Description

Sets/Gets the basic frequency range offset to get the receiver frequency when the frequency offset feature is ON and offset type is "SRCRcv".

The value is expressed in hertz (Hz).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FrequencyReceiverOffset As Integer
```

C#

```
public static int FrequencyReceiverOffset
```

Visual C++

```
public:  
static int FrequencyReceiverOffset
```

JavaScript

```
CMT.Instruments.Attributes.frequencyReceiverOffset
```

Back to [Attributes](#)

FrequencyReceiverOffsetStart

Description

Sets/Gets the frequency sweep start of the receivers when the frequency offset feature is ON and offset type is "SRCRCv".

The value is expressed in hertz (Hz).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FrequencyReceiverOffsetStart As Integer
```

C#

```
public static int FrequencyReceiverOffsetStart
```

Visual C++

```
public:  
static int FrequencyReceiverOffsetStart
```

JavaScript

```
CMT.Instruments.Attributes.frequencyReceiverOffsetStart
```

Back to [Attributes](#)

FrequencyReceiverOffsetStop

Description

Sets/Gets the frequency sweep stop of the receivers when the frequency offset feature is ON and offset type is "SRCRCv".

The value is expressed in hertz (Hz).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FrequencyReceiverOffsetStop As Integer
```

C#

```
public static int FrequencyReceiverOffsetStop
```

Visual C++

```
public:  
static int FrequencyReceiverOffsetStop
```

JavaScript

```
CMT.Instruments.Attributes.frequencyReceiverOffsetStop
```

Back to [Attributes](#)

FrequencySourceDivider

Description

Sets/Gets the basic frequency range divisor to get the source frequency when the frequency offset feature is ON and offset type is "SRCRcv".

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FrequencySourceDivider As Integer
```

C#

```
public static int FrequencySourceDivider
```

Visual C++

```
public:  
static int FrequencySourceDivider
```

JavaScript

```
CMT.Instruments.Attributes.frequencySourceDivider
```

Back to [Attributes](#)

FrequencySourceMultiplier

Description

Sets/Gets the basic frequency range multiplier to get the source frequency when the frequency offset feature is ON and offset type is "SRCRcv".

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FrequencySourceMultiplier As Integer
```

C#

```
public static int FrequencySourceMultiplier
```

Visual C++

```
public:  
static int FrequencySourceMultiplier
```

JavaScript

```
CMT.Instruments.Attributes.frequencySourceMultiplier
```

Back to [Attributes](#)

FrequencySourceOffset

Description

Sets/Gets the basic frequency range offset to get the source frequency when the frequency offset feature is ON and offset type is "SRCRcv".

The value is expressed in hertz (Hz).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FrequencySourceOffset As Integer
```

C#

```
public static int FrequencySourceOffset
```

Visual C++

```
public:  
static int FrequencySourceOffset
```

JavaScript

```
CMT.Instruments.Attributes.frequencySourceOffset
```

Back to [Attributes](#)

FrequencySourceOffsetStart

Description

Sets/Gets the frequency sweep start of the source when the frequency offset feature is ON and offset type is "SRCRcv".

The value is expressed in hertz (Hz).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FrequencySourceOffsetStart As Integer
```

C#

```
public static int FrequencySourceOffsetStart
```

Visual C++

```
public:  
static int FrequencySourceOffsetStart
```

JavaScript

```
CMT.Instruments.Attributes.frequencySourceOffsetStart
```

Back to [Attributes](#)

FrequencySourceOffsetStop

Description

Sets/Gets the frequency sweep stop of the source when the frequency offset feature is ON and offset type is "SRCRCv".

The value is expressed in hertz (Hz).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FrequencySourceOffsetStop As Integer
```

C#

```
public static int FrequencySourceOffsetStop
```

Visual C++

```
public:  
static int FrequencySourceOffsetStop
```

JavaScript

```
CMT.Instruments.Attributes.frequencySourceOffsetStop
```

Back to [Attributes](#)

FunctionDomainCoupling

Description

If the arbitrary range is turned ON by [FunctionDomainState](#), it specifies whether all traces of repCapID use the same range (coupling) or each trace uses individual range when [FunctionExecute](#) is executed.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FunctionDomainCoupling As Integer
```

C#

```
public static int FunctionDomainCoupling
```

Visual C++

```
public:  
static int FunctionDomainCoupling
```

JavaScript

```
CMT.Instruments.Attributes.functionDomainCoupling
```

Back to [Attributes](#)

FunctionDomainStart

Description

Sets/Gets the start value of the analysis range of [FunctionExecute](#) function.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FunctionDomainStart As Integer
```

C#

```
public static int FunctionDomainStart
```

Visual C++

```
public:  
static int FunctionDomainStart
```

JavaScript

```
CMT.Instruments.Attributes.functionDomainStart
```

Back to [Attributes](#)

FunctionDomainState

Description

Specifies whether an arbitrary range or the entire sweep range is used when [FunctionExecute](#) is executed.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FunctionDomainState As Integer
```

C#

```
public static int FunctionDomainState
```

Visual C++

```
public:  
static int FunctionDomainState
```

JavaScript

```
CMT.Instruments.Attributes.functionDomainState
```

Back to [Attributes](#)

FunctionDomainStop

Description

Sets/Gets the stop value of the analysis range of [FunctionExecute](#) function.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FunctionDomainStop As Integer
```

C#

```
public static int FunctionDomainStop
```

Visual C++

```
public:  
static int FunctionDomainStop
```

JavaScript

```
CMT.Instruments.Attributes.functionDomainStop
```

Back to [Attributes](#)

FunctionPeakExcursion

Description

Sets/Gets the lower limit for the peak excursion value when executing the peak search with [FunctionExecute](#) function.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FunctionPeakExcursion As Integer
```

C#

```
public static int FunctionPeakExcursion
```

Visual C++

```
public:  
static int FunctionPeakExcursion
```

JavaScript

```
CMT.Instruments.Attributes.functionPeakExcursion
```

Back to [Attributes](#)

FunctionPeakPolarity

Description

Sets/Gets the polarity when performing the peak search with [FunctionExecute](#) function.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[FunctionPeakPolarityBoth](#)

[FunctionPeakPolarityNegative](#)

[FunctionPeakPolarityPositive](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FunctionPeakPolarity As Integer
```

C#

```
public static int FunctionPeakPolarity
```

Visual C++

```
public:
```

Visual C++

```
static int FunctionPeakPolarity
```

JavaScript

```
CMT.Instruments.Attributes.functionPeakPolarity
```

Back to [Attributes](#)

FunctionPoints

Description

Gets the number of points (data pairs) of the analysis result by [FunctionExecute](#) function.

Always reads out 1, when the search is executed for the maximum, minimum, mean, standard deviation, peak, and peak-to-peak values.

The actual number of points is read out, when the search is executed for all peak or all targets.

Used as attributeID with functions: [GetAttributeViInt32](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FunctionPoints As Integer
```

C#

```
public static int FunctionPoints
```

Visual C++

```
public:  
static int FunctionPoints
```

JavaScript

```
CMT.Instruments.Attributes.functionPoints
```

Back to [Attributes](#)

FunctionTargetLevel

Description

Sets/Gets the target level when performing the search for the trace and the target level crosspoints with [FunctionExecute](#) function.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FunctionTargetLevel As Integer
```

C#

```
public static int FunctionTargetLevel
```

Visual C++

```
public:  
static int FunctionTargetLevel
```

JavaScript

```
CMT.Instruments.Attributes.functionTargetLevel
```

Back to [Attributes](#)

FunctionTransitionType

Description

Sets/Gets the transition type when performing the search for the trace and the target level crosspoints with [FunctionExecute](#) function.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[FunctionTransitionTypeBoth](#)

[FunctionTransitionTypeNegative](#)

[FunctionTransitionTypePositive](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FunctionTransitionType As Integer
```

C#

```
public static int FunctionTransitionType
```

Visual C++

```
public:
```

Visual C++

`static int FunctionTransitionType`

JavaScript

`CMT.Instruments.Attributes.functionTransitionType`

Back to [Attributes](#)

FunctionType

Description

Sets/Gets the type of analysis executed by [FunctionExecute](#) function.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[FunctionAllPeaks](#)

[FunctionAllTarget](#)

[FunctionMaximum](#)

[FunctionMean](#)

[FunctionMinimum](#)

[FunctionPeak](#)

[FunctionPeakToPeak](#)

[FunctionStandardDeviation](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FunctionType As Integer
```

C#

```
public static int FunctionType
```

Visual C++

```
public:  
static int FunctionType
```

JavaScript

```
CMT.Instruments.Attributes.functionType
```

Back to [Attributes](#)

LimitLineResponseOffset

Description

Sets/Gets the value of the limit line offset along Y-axis.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared LimitLineResponseOffset As Integer
```

C#

```
public static int LimitLineResponseOffset
```

Visual C++

```
public:  
static int LimitLineResponseOffset
```

JavaScript

```
CMT.Instruments.Attributes.limitLineResponseOffset
```

Back to [Attributes](#)

LimitLineStimulusOffset

Description

Sets/Gets the value of the limit line offset along X-axis.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared LimitLineStimulusOffset As Integer
```

C#

```
public static int LimitLineStimulusOffset
```

Visual C++

```
public:  
static int LimitLineStimulusOffset
```

JavaScript

```
CMT.Instruments.Attributes.limitLineStimulusOffset
```

Back to [Attributes](#)

LogicalName

Description

Logical Name identifies a driver session in the Configuration Store.

If Logical Name is not empty, the driver was initialized from information in the driver session.

If it is empty, the driver was initialized without using the Configuration Store.

Used as attributeID with functions: [GetAttributeViString](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared LogicalName As Integer
```

C#

```
public static int LogicalName
```

Visual C++

```
public:  
static int LogicalName
```

JavaScript

```
CMT.Instruments.Attributes.logicalName
```

Back to [Attributes](#)

MarkerMathBandwidthSearch

Description

Turns ON/OFF the bandwidth search function.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerMathBandwidthSearch As Integer
```

C#

```
public static int MarkerMathBandwidthSearch
```

Visual C++

```
public:  
static int MarkerMathBandwidthSearch
```

JavaScript

```
CMT.Instruments.Attributes.markerMathBandwidthSearch
```

Back to [Attributes](#)

MarkerMathBandwidthSearchRef

Description

Selects the reference point for the bandwidth search function: reference marker or absolute maximum value of the trace.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerMathBandwidthSearchRef As Integer
```

C#

```
public static int MarkerMathBandwidthSearchRef
```

Visual C++

```
public:  
static int MarkerMathBandwidthSearchRef
```

JavaScript

```
CMT.Instruments.Attributes.markerMathBandwidthSearchRef
```

Back to [Attributes](#)

MarkerMathBandwidthSearchType

Description

Sets/Gets the type of the bandwidth search function.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[MarkerMathBandwidthSearchBandpass](#)

[MarkerMathBandwidthSearchNotch](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerMathBandwidthSearchType As Integer
```

C#

```
public static int MarkerMathBandwidthSearchType
```

Visual C++

```
public:  
static int MarkerMathBandwidthSearchType
```

JavaScript

CMT.Instruments.Attributes.markerMathBandwidthSearchType

Back to [Attributes](#)

MarkerMathBandwidthSearchValue

Description

Sets/Gets the bandwidth definition value.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Marker index for the active trace of channel i.e. "Channel1:Marker2" and so on. The physical names are supported along with the corresponding Channel index and Measurement index and Marker index for the n-th trace of channel i.e. "Channel1:Measurement3:Marker2" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerMathBandwidthSearchValue As Integer
```

C#

```
public static int MarkerMathBandwidthSearchValue
```

Visual C++

```
public:  
static int MarkerMathBandwidthSearchValue
```

JavaScript

```
CMT.Instruments.Attributes.markerMathBandwidthSearchValue
```

Back to [Attributes](#)

MarkerMathFlatness

Description

Turns ON/OFF the marker FLATNESS function.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerMathFlatness As Integer
```

C#

```
public static int MarkerMathFlatness
```

Visual C++

```
public:  
static int MarkerMathFlatness
```

JavaScript

```
CMT.Instruments.Attributes.markerMathFlatness
```

Back to [Attributes](#)

MarkerMathFlatnessStart

Description

Sets/Gets the number of the marker, which specifies the start frequency of the FLATNESS function domain.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerMathFlatnessStart As Integer
```

C#

```
public static int MarkerMathFlatnessStart
```

Visual C++

```
public:  
static int MarkerMathFlatnessStart
```

JavaScript

```
CMT.Instruments.Attributes.markerMathFlatnessStart
```

Back to [Attributes](#)

MarkerMathFlatnessStop

Description

Sets/Gets the number of the marker, which specifies the stop frequency of the FLATNESS function domain.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerMathFlatnessStop As Integer
```

C#

```
public static int MarkerMathFlatnessStop
```

Visual C++

```
public:  
static int MarkerMathFlatnessStop
```

JavaScript

```
CMT.Instruments.Attributes.markerMathFlatnessStop
```

Back to [Attributes](#)

MarkerMathStatisticRange

Description

Sets/Gets either partial frequency range or entire frequency range is used for math statistic calculation.

The partial frequency range is limited by two markers.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerMathStatisticRange As Integer
```

C#

```
public static int MarkerMathStatisticRange
```

Visual C++

```
public:  
static int MarkerMathStatisticRange
```

JavaScript

```
CMT.Instruments.Attributes.markerMathStatisticRange
```

Back to [Attributes](#)

MarkerMathStatistics

Description

Turns ON/OFF the math statistics display.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerMathStatistics As Integer
```

C#

```
public static int MarkerMathStatistics
```

Visual C++

```
public:  
static int MarkerMathStatistics
```

JavaScript

```
CMT.Instruments.Attributes.markerMathStatistics
```

Back to [Attributes](#)

MarkerMathStatisticStart

Description

Sets/Gets the number of the marker, which specifies the start frequency of the math statistics range.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerMathStatisticStart As Integer
```

C#

```
public static int MarkerMathStatisticStart
```

Visual C++

```
public:  
static int MarkerMathStatisticStart
```

JavaScript

```
CMT.Instruments.Attributes.markerMathStatisticStart
```

Back to [Attributes](#)

MarkerMathStatisticStop

Description

Sets/Gets the number of the marker, which specifies the stop frequency of the math statistics range.

Used as attributeID with functions:

[SetAttributeVInt32](#)

[GetAttributeVInt32](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerMathStatisticStop As Integer
```

C#

```
public static int MarkerMathStatisticStop
```

Visual C++

```
public:  
static int MarkerMathStatisticStop
```

JavaScript

```
CMT.Instruments.Attributes.markerMathStatisticStop
```

Back to [Attributes](#)

MarkerPropertiesAlign

Description

Sets/Gets the alignment mode of the marker display position of each trace, when the only active trace display feature is turned OFF by [MarkerPropertiesMarkerActiveOnly](#).

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerPropertiesAlign As Integer
```

C#

```
public static int MarkerPropertiesAlign
```

Visual C++

```
public:  
static int MarkerPropertiesAlign
```

JavaScript

```
CMT.Instruments.Attributes.markerPropertiesAlign
```

Back to [Attributes](#)

MarkerPropertiesDataXPosition

Description

Sets/Gets the display position of the marker annotation on the X-axis by a percentage of the display width. The value is expressed in percentages (%).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Measurement index i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerPropertiesDataXPosition As Integer
```

C#

```
public static int MarkerPropertiesDataXPosition
```

Visual C++

```
public:  
static int MarkerPropertiesDataXPosition
```

JavaScript

```
CMT.Instruments.Attributes.markerPropertiesDataXPosition
```

Back to [Attributes](#)

MarkerPropertiesDataYPosition

Description

Sets/Gets the display position of the marker annotation on the Y-axis by a percentage of the display height.

The value is expressed in percentages (%).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Measurement index i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerPropertiesDataYPosition As Integer
```

C#

```
public static int MarkerPropertiesDataYPosition
```

Visual C++

```
public:  
static int MarkerPropertiesDataYPosition
```

JavaScript

```
CMT.Instruments.Attributes.markerPropertiesDataYPosition
```

Back to [Attributes](#)

MarkerPropertiesDiscrete

Description

Turns ON/OFF the marker discrete mode.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerPropertiesDiscrete As Integer
```

C#

```
public static int MarkerPropertiesDiscrete
```

Visual C++

```
public:  
static int MarkerPropertiesDiscrete
```

JavaScript

```
CMT.Instruments.Attributes.markerPropertiesDiscrete
```

Back to [Attributes](#)

MarkerPropertiesMarkerActiveOnly

Description

Sets/Gets display either the active trace markers or all trace markers.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerPropertiesMarkerActiveOnly As Integer
```

C#

```
public static int MarkerPropertiesMarkerActiveOnly
```

Visual C++

```
public:  
static int MarkerPropertiesMarkerActiveOnly
```

JavaScript

```
CMT.Instruments.Attributes.markerPropertiesMarkerActiveOnly
```

Back to [Attributes](#)

MarkerPropertiesMarkerCouple

Description

Turns ON/OFF the marker coupling between traces.

When coupled the markers of different traces with same number track the X-axis position.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerPropertiesMarkerCouple As Integer
```

C#

```
public static int MarkerPropertiesMarkerCouple
```

Visual C++

```
public:  
static int MarkerPropertiesMarkerCouple
```

JavaScript

```
CMT.Instruments.Attributes.markerPropertiesMarkerCouple
```

Back to [Attributes](#)

MarkerPropertiesMarkerTable

Description

Turns ON/OFF of the marker table.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerPropertiesMarkerTable As Integer
```

C#

```
public static int MarkerPropertiesMarkerTable
```

Visual C++

```
public:  
static int MarkerPropertiesMarkerTable
```

JavaScript

```
CMT.Instruments.Attributes.markerPropertiesMarkerTable
```

Back to [Attributes](#)

MarkersCount

Description

Sets\Gets the number of the turned ON markers.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkersCount As Integer
```

C#

```
public static int MarkersCount
```

Visual C++

```
public:  
static int MarkersCount
```

JavaScript

```
CMT.Instruments.Attributes.markersCount
```

Back to [Attributes](#)

MarkerSearchCouple

Description

If the arbitrary search range is turned ON by [MarkerSearchRange](#), specifies whether all traces of repCapID use the same range (coupling) or each trace uses individual range when the marker search is performed.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerSearchCouple As Integer
```

C#

```
public static int MarkerSearchCouple
```

Visual C++

```
public:  
static int MarkerSearchCouple
```

JavaScript

```
CMT.Instruments.Attributes.markerSearchCouple
```

Back to [Attributes](#)

MarkerSearchPeakExcursion

Description

Sets/Gets the start value of the marker search range.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Marker index for the active trace of channel i.e. "Channel1:Marker2" and so on. The physical names are supported along with the corresponding Channel index and Measurement index and Marker index for the n-th trace of channel i.e. "Channel1:Measurement3:Marker2" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerSearchPeakExcursion As Integer
```

C#

```
public static int MarkerSearchPeakExcursion
```

Visual C++

```
public:  
static int MarkerSearchPeakExcursion
```

JavaScript

```
CMT.Instruments.Attributes.markerSearchPeakExcursion
```

Back to [Attributes](#)

MarkerSearchPeakPolarity

Description

Sets/Gets the peak polarity, when the marker search for peak is performed by [MarkerFunctionExecute](#) function.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[MarkerSearchPeakPolarityBoth](#)

[MarkerSearchPeakPolarityNegative](#)

[MarkerSearchPeakPolarityPositive](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Marker index for the active trace of channel i.e. "Channel1:Marker2" and so on. The physical names are supported along with the corresponding Channel index and Measurement index and Marker index for the n-th trace of channel i.e. "Channel1:Measurement3:Marker2" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerSearchPeakPolarity As Integer
```

C#

```
public static int MarkerSearchPeakPolarity
```

Visual C++

```
public:
```

Visual C++

```
static int MarkerSearchPeakPolarity
```

JavaScript

```
CMT.Instruments.Attributes.markerSearchPeakPolarity
```

Back to [Attributes](#)

MarkerSearchRange

Description

Specifies whether an arbitrary range or the entire sweep range is used when the marker search is performed.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerSearchRange As Integer
```

C#

```
public static int MarkerSearchRange
```

Visual C++

```
public:  
static int MarkerSearchRange
```

JavaScript

```
CMT.Instruments.Attributes.markerSearchRange
```

Back to [Attributes](#)

MarkerSearchStart

Description

Sets/Gets the start value of the marker search range.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerSearchStart As Integer
```

C#

```
public static int MarkerSearchStart
```

Visual C++

```
public:  
static int MarkerSearchStart
```

JavaScript

```
CMT.Instruments.Attributes.markerSearchStart
```

Back to [Attributes](#)

MarkerSearchStop

Description

Sets/Gets the stop value of the marker search range.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerSearchStop As Integer
```

C#

```
public static int MarkerSearchStop
```

Visual C++

```
public:  
static int MarkerSearchStop
```

JavaScript

```
CMT.Instruments.Attributes.markerSearchStop
```

Back to [Attributes](#)

MarkerSearchTargetTransition

Description

Sets/Gets the type of the target transition, when the marker search for transition is performed by [MarkerFunctionExecute](#) function.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[MarkerSearchTargetTransitionBoth](#)

[MarkerSearchTargetTransitionNegative](#)

[MarkerSearchTargetTransitionPositive](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Marker index for the active trace of channel i.e. "Channel1:Marker2" and so on. The physical names are supported along with the corresponding Channel index and Measurement index and Marker index for the n-th trace of channel i.e. "Channel1:Measurement3:Marker2" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerSearchTargetTransition As Integer
```

C#

```
public static int MarkerSearchTargetTransition
```

Visual C++

```
public:
```

Visual C++

```
static int MarkerSearchTargetTransition
```

JavaScript

```
CMT.Instruments.Attributes.markerSearchTargetTransition
```

Back to [Attributes](#)

MarkerSearchTargetValue

Description

Sets/Gets the target value, when the marker search for target is performed by [MarkerFunctionExecute](#) function.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Marker index for the active trace of channel i.e. "Channel1:Marker2" and so on. The physical names are supported along with the corresponding Channel index and Measurement index and Marker index for the n-th trace of channel i.e. "Channel1:Measurement3:Marker2" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerSearchTargetValue As Integer
```

C#

```
public static int MarkerSearchTargetValue
```

Visual C++

```
public:  
static int MarkerSearchTargetValue
```

JavaScript

```
CMT.Instruments.Attributes.markerSearchTargetValue
```

Back to [Attributes](#)

MarkerSearchTracking

Description

Turns ON/OFF the marker search tracking.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Marker index for the active trace of channel i.e. "Channel1:Marker2" and so on. The physical names are supported along with the corresponding Channel index and Measurement index and Marker index for the n-th trace of channel i.e. "Channel1:Measurement3:Marker2" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerSearchTracking As Integer
```

C#

```
public static int MarkerSearchTracking
```

Visual C++

```
public:  
static int MarkerSearchTracking
```

JavaScript

```
CMT.Instruments.Attributes.markerSearchTracking
```

Back to [Attributes](#)

MarkerSearchType

Description

Sets the type of the marker search, which is performed by [MarkerFunctionExecute](#) function.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[MarkerSearchTypeMaximum](#)

[MarkerSearchTypeMinimum](#)

[MarkerSearchTypePeak](#)

[MarkerSearchTypePeakLeft](#)

[MarkerSearchTypePeakRight](#)

[MarkerSearchTypeTarget](#)

[MarkerSearchTypeTargetLeft](#)

[MarkerSearchTypeTargetRight](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Marker index for the active trace of channel i.e. "Channel1:Marker2" and so on. The physical names are supported along with the corresponding Channel index and Measurement index and Marker index for the n–th trace of channel i.e. "Channel1:Measurement3:Marker2" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerSearchType As Integer
```

C#

```
public static int MarkerSearchType
```

Visual C++

```
public:  
static int MarkerSearchType
```

JavaScript

```
CMT.Instruments.Attributes.markerSearchType
```

Back to [Attributes](#)

MarkerStimulus

Description

Sets/Gets the stimulus value of the marker.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Marker index for the active trace of channel i.e. "Channel1:Marker2" and so on. The physical names are supported along with the corresponding Channel index and Measurement index and Marker index for the n-th trace of channel i.e. "Channel1:Measurement3:Marker2" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerStimulus As Integer
```

C#

```
public static int MarkerStimulus
```

Visual C++

```
public:  
static int MarkerStimulus
```

JavaScript

```
CMT.Instruments.Attributes.markerStimulus
```

Back to [Attributes](#)

MeasurementAveraging

Description

Turns ON/OFF the measurement averaging function.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementAveraging As Integer
```

C#

```
public static int MeasurementAveraging
```

Visual C++

```
public:  
static int MeasurementAveraging
```

JavaScript

```
CMT.Instruments.Attributes.measurementAveraging
```

Back to [Attributes](#)

MeasurementAveragingFactor

Description

Sets/Gets the averaging factor, when the averaging function is turned on.

Used as attributeID with functions:

[SetAttributeVInt32](#)

[GetAttributeVInt32](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementAveragingFactor As Integer
```

C#

```
public static int MeasurementAveragingFactor
```

Visual C++

```
public:  
static int MeasurementAveragingFactor
```

JavaScript

```
CMT.Instruments.Attributes.measurementAveragingFactor
```

Back to [Attributes](#)

MeasurementAvgTrigger

Description

Turns ON/OFF the averaging trigger function.

The function executes a sweep the number of times specified by the averaging factor with a single trigger for the channels with the averaging enabled.

The averaging process begins again with each trigger.

NOTE

The point trigger function has priority against this command. When the point trigger is enabled the number of pulses equal to (number of points) x (averaging factor) is needed to complete the averaging.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementAvgTrigger As Integer
```

C#

```
public static int MeasurementAvgTrigger
```

Visual C++

```
public:
```

Visual C++

```
static int MeasurementAvgTrigger
```

JavaScript

```
CMT.Instruments.Attributes.measurementAvgTrigger
```

Back to [Attributes](#)

MeasurementDivisions

Description

Sets/Gets the number of the vertical scale divisions.

For the rectangular format only.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementDivisions As Integer
```

C#

```
public static int MeasurementDivisions
```

Visual C++

```
public:  
static int MeasurementDivisions
```

JavaScript

```
CMT.Instruments.Attributes.measurementDivisions
```

Back to [Attributes](#)

MeasurementElecDelay

Description

Sets/Gets the value of the electrical delay. The value is expressed in seconds (sec).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementElecDelay As Integer
```

C#

```
public static int MeasurementElecDelay
```

Visual C++

```
public:  
static int MeasurementElecDelay
```

JavaScript

```
CMT.Instruments.Attributes.measurementElecDelay
```

Back to [Attributes](#)

MeasurementFormat

Description

Sets/Gets the display format specified by Measurement Format Enum for the measurement.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[MeasurementFormatExpandPhase](#)

[MeasurementFormatGroupDelay](#)

[MeasurementFormatImag](#)

[MeasurementFormatLinMag](#)

[MeasurementFormatLogMag](#)

[MeasurementFormatPhase](#)

[MeasurementFormatPolarLin](#)

[MeasurementFormatPolarLog](#)

[MeasurementFormatPolarReallImage](#)

[MeasurementFormatReal](#)

[MeasurementFormatSmithAdmittance](#)

[MeasurementFormatSmithComplex](#)

[MeasurementFormatSmithLin](#)

[MeasurementFormatSmithLog](#)

[MeasurementFormatSmithReallImage](#)

[MeasurementFormatSwr](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementFormat As Integer
```

C#

```
public static int MeasurementFormat
```

Visual C++

```
public:  
static int MeasurementFormat
```

JavaScript

```
CMT.Instruments.Attributes.measurementFormat
```

Back to [Attributes](#)

MeasurementPhaseOffset

Description

Sets/Gets the value of the phase offset. The value is expressed in degrees.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementPhaseOffset As Integer
```

C#

```
public static int MeasurementPhaseOffset
```

Visual C++

```
public:  
static int MeasurementPhaseOffset
```

JavaScript

```
CMT.Instruments.Attributes.measurementPhaseOffset
```

Back to [Attributes](#)

MeasurementRefPosition

Description

Sets/Gets the position of the reference line.

For the rectangular format only.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Measurement index i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementRefPosition As Integer
```

C#

```
public static int MeasurementRefPosition
```

Visual C++

```
public:  
static int MeasurementRefPosition
```

JavaScript

```
CMT.Instruments.Attributes.measurementRefPosition
```

Back to [Attributes](#)

MeasurementRefValue

Description

Sets/Gets the value of the reference line (response value on the reference line).

For the rectangular format only.

The value is depending on the format.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Measurement index i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementRefValue As Integer
```

C#

```
public static int MeasurementRefValue
```

Visual C++

```
public:  
static int MeasurementRefValue
```

JavaScript

```
CMT.Instruments.Attributes.measurementRefValue
```

Back to [Attributes](#)

MeasurementScaleDiv

Description

Sets/Gets the trace scale. Sets the scale per division, when the data format is the rectangular format.

Sets the full scale value, when the data format is the Smith chart format or the polar format.

The value is depending on the format.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Measurement index i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementScaleDiv As Integer
```

C#

```
public static int MeasurementScaleDiv
```

Visual C++

```
public:  
static int MeasurementScaleDiv
```

JavaScript

CMT.Instruments.Attributes.measurementScaleDiv

Back to [Attributes](#)

MeasurementSmoothing

Description

Turns ON/OFF the trace smoothing.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementSmoothing As Integer
```

C#

```
public static int MeasurementSmoothing
```

Visual C++

```
public:  
static int MeasurementSmoothing
```

JavaScript

```
CMT.Instruments.Attributes.measurementSmoothing
```

Back to [Attributes](#)

MeasurementSmoothingAperture

Description

Sets/Gets the smoothing aperture, when performing smoothing function. The value is expressed in percentages (%).

Used as attributeID with functions:

[SetAttributeVInt32](#)

[GetAttributeVInt32](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementSmoothingAperture As Integer
```

C#

```
public static int MeasurementSmoothingAperture
```

Visual C++

```
public:  
static int MeasurementSmoothingAperture
```

JavaScript

```
CMT.Instruments.Attributes.measurementSmoothingAperture
```

Back to [Attributes](#)

NumberOfPorts

Description

Gets the number of the ports.

Used as attributeID with function [GetAttributeViInt32](#).

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared NumberOfPorts As Integer
```

C#

```
public static int NumberOfPorts
```

Visual C++

```
public:  
static int NumberOfPorts
```

JavaScript

```
CMT.Instruments.Attributes.numberOfPorts
```

Back to [Attributes](#)

OutputTriggerRoute

NOTE

Command valid for PXIe-S5090 model only

Description

Sets/Gets the connector to use for the trigger output in a PXI system.

One of the 9 routes can be selected. The same line cannot be selected as input and output trigger route.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[TriggerRouteSmb](#)

[TriggerRoutePxiTrig0](#)

[TriggerRoutePxiTrig1](#)

[TriggerRoutePxiTrig2](#)

[TriggerRoutePxiTrig3](#)

[TriggerRoutePxiTrig4](#)

[TriggerRoutePxiTrig5](#)

[TriggerRoutePxiTrig6](#)

[TriggerRoutePxiTrig7](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared OutputTriggerRoute As Integer
```

C#

```
public static int OutputTriggerRoute
```

Visual C++

```
public:  
static int OutputTriggerRoute
```

JavaScript

```
CMT.Instruments.Attributes.outputTriggerRoute
```

Back to [Attributes](#)

PortExtensions

Description

Turns ON/OFF the port extension function.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared PortExtensions As Integer
```

C#

```
public static int PortExtensions
```

Visual C++

```
public:  
static int PortExtensions
```

JavaScript

```
CMT.Instruments.Attributes.portExtensions
```

Back to [Attributes](#)

PortExtensionsFreq1Value

Description

Sets/Gets the values of the frequency 1 to calculate the loss for the port extension function. The value is expressed in Hertz (Hz).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared PortExtensionsFreq1Value As Integer
```

C#

```
public static int PortExtensionsFreq1Value
```

Visual C++

```
public:  
static int PortExtensionsFreq1Value
```

JavaScript

```
CMT.Instruments.Attributes.portExtensionsFreq1Value
```

Back to [Attributes](#)

PortExtensionsFreq2Value

Description

Sets/Gets the values of the frequency 2 to calculate the loss for the port extension function. The value is expressed in Hertz (Hz).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared PortExtensionsFreq2Value As Integer
```

C#

```
public static int PortExtensionsFreq2Value
```

Visual C++

```
public:  
static int PortExtensionsFreq2Value
```

JavaScript

```
CMT.Instruments.Attributes.portExtensionsFreq2Value
```

Back to [Attributes](#)

PortExtensionsLdcValue

Description

Sets/Gets the loss value at DC for the port extension function. The value is expressed in decibels (dB).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared PortExtensionsLdcValue As Integer
```

C#

```
public static int PortExtensionsLdcValue
```

Visual C++

```
public:  
static int PortExtensionsLdcValue
```

JavaScript

```
CMT.Instruments.Attributes.portExtensionsLdcValue
```

Back to [Attributes](#)

PortExtensionsLoss1State

Description

Turns ON/OFF the loss compensation of the loss 1 for the port extension function.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared PortExtensionsLoss1State As Integer
```

C#

```
public static int PortExtensionsLoss1State
```

Visual C++

```
public:  
static int PortExtensionsLoss1State
```

JavaScript

```
CMT.Instruments.Attributes.portExtensionsLoss1State
```

Back to [Attributes](#)

PortExtensionsLoss1Value

Description

Sets/Gets the values of the loss 1 for the port extension function. The value is expressed in decibels (dB).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared PortExtensionsLoss1Value As Integer
```

C#

```
public static int PortExtensionsLoss1Value
```

Visual C++

```
public:  
static int PortExtensionsLoss1Value
```

JavaScript

```
CMT.Instruments.Attributes.portExtensionsLoss1Value
```

Back to [Attributes](#)

PortExtensionsLoss2State

Description

Turns ON/OFF the loss compensation of the loss 2 for the port extension function.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared PortExtensionsLoss2State As Integer
```

C#

```
public static int PortExtensionsLoss2State
```

Visual C++

```
public:  
static int PortExtensionsLoss2State
```

JavaScript

```
CMT.Instruments.Attributes.portExtensionsLoss2State
```

Back to [Attributes](#)

PortExtensionsLoss2Value

Description

Sets/Gets the values of the loss 2 for the port extension function. The value is expressed in decibels (dB).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared PortExtensionsLoss2Value As Integer
```

C#

```
public static int PortExtensionsLoss2Value
```

Visual C++

```
public:  
static int PortExtensionsLoss2Value
```

JavaScript

```
CMT.Instruments.Attributes.portExtensionsLoss2Value
```

Back to [Attributes](#)

PortExtensionsTime

Description

Sets/Gets the electrical delay value for the port extension function. The value is expressed in seconds (sec).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared PortExtensionsTime As Integer
```

C#

```
public static int PortExtensionsTime
```

Visual C++

```
public:  
static int PortExtensionsTime
```

JavaScript

```
CMT.Instruments.Attributes.portExtensionsTime
```

Back to [Attributes](#)

PowerCalibrationCorrection

Description

Turns ON/OFF the power correction.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared PowerCalibrationCorrection As Integer
```

C#

```
public static int PowerCalibrationCorrection
```

Visual C++

```
public:  
static int PowerCalibrationCorrection
```

JavaScript

```
CMT.Instruments.Attributes.powerCalibrationCorrection
```

Back to [Attributes](#)

PowerCalibrationLossCompensation

Description

Turns ON/OFF the state of the loss compensation used when the power calibration is executed by [TakePowerCalSweep](#) function.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port2" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared PowerCalibrationLossCompensation As Integer
```

C#

```
public static int PowerCalibrationLossCompensation
```

Visual C++

```
public:  
static int PowerCalibrationLossCompensation
```

JavaScript

```
CMT.Instruments.Attributes.powerCalibrationLossCompensation
```

Back to [Attributes](#)

PowerSensorReady

Description

Gets the ready state of the power sensor.

The state is True when the power sensor is ready.

Used as attributeID with function [GetAttributeViBoolean](#).

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared PowerSensorReady As Integer
```

C#

```
public static int PowerSensorReady
```

Visual C++

```
public:  
static int PowerSensorReady
```

JavaScript

```
CMT.Instruments.Attributes.powerSensorReady
```

Back to [Attributes](#)

PowerSensorType

Description

Sets/Gets the power sensor type to be used in a source power calibration.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[PowerSensorNrpxt](#)

[PowerSensorNrpxz](#)

[PowerSensorNrpxs](#)

[PowerSensorU200x](#)

[PowerSensorU848x](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared PowerSensorType As Integer
```

C#

```
public static int PowerSensorType
```

Visual C++

```
public:
```


Visual C++

```
static int PowerSensorType
```

JavaScript

```
CMT.Instruments.Attributes.powerSensorType
```

Back to [Attributes](#)

PowerTripAtOverload

Description

Turns ON/OFF the Power Trip at Overload function. Except for Planar-804/808/304 Models.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared PowerTripAtOverload As Integer
```

C#

```
public static int PowerTripAtOverload
```

Visual C++

```
public:  
static int PowerTripAtOverload
```

JavaScript

```
CMT.Instruments.Attributes.powerTripAtOverload
```

Back to [Attributes](#)

PrintDateAndTime

Description

Turns ON/OFF the date and time printout in the upper right corner of the image.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared PrintDateAndTime As Integer
```

C#

```
public static int PrintDateAndTime
```

Visual C++

```
public:  
static int PrintDateAndTime
```

JavaScript

```
CMT.Instruments.Attributes.printDateAndTime
```

Back to [Attributes](#)

PrintInvertImage

Description

Turns ON/OFF the inverted color image printout.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared PrintInvertImage As Integer
```

C#

```
public static int PrintInvertImage
```

Visual C++

```
public:  
static int PrintInvertImage
```

JavaScript

```
CMT.Instruments.Attributes.printInvertImage
```

Back to [Attributes](#)

ReferenceFrequencySource

Description

Sets/Gets the internal or external source of the reference frequency of 10 MHz.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[ReferenceFrequencySourceExternal](#)

[ReferenceFrequencySourceInternal](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ReferenceFrequencySource As Integer
```

C#

```
public static int ReferenceFrequencySource
```

Visual C++

```
public:  
static int ReferenceFrequencySource
```

JavaScript

CMT.Instruments.Attributes.referenceFrequencySource

Back to [Attributes](#)

ReferenceMarker

Description

Turns ON/OFF the reference marker.

When the reference marker is turned ON, all the values of the other markers turn to relative values.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ReferenceMarker As Integer
```

C#

```
public static int ReferenceMarker
```

Visual C++

```
public:  
static int ReferenceMarker
```

JavaScript

```
CMT.Instruments.Attributes.referenceMarker
```

Back to [Attributes](#)

SaveTouchstoneFileColumnSeparator

Description

Sets/Gets the Touchstone file separator symbol when the S-parameters are saved by [SaveTouchstoneFile](#) function.

The attribute is used for the active channel.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[SaveTouchstoneFileColumnSeparatorSpace](#)

[SaveTouchstoneFileColumnSeparatorTab](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared SaveTouchstoneFileColumnSeparator As Integer
```

C#

```
public static int SaveTouchstoneFileColumnSeparator
```

Visual C++

```
public:  
static int SaveTouchstoneFileColumnSeparator
```

JavaScript

CMT.Instruments.Attributes.saveTouchstoneFileColumnSeparator

Back to [Attributes](#)

SaveTouchstoneFileFormat

Description

Sets/Gets the data format for the S-parameter saving by [SaveTouchstoneFile](#) function.

The attribute is used for the active channel.

Used as attributeID with functions:

[SetAttributeVInt32](#)

[GetAttributeVInt32](#)

Field Value

[SaveTouchstoneFileFormatDbAngle](#)

[SaveTouchstoneFileFormatMagnitudeAngle](#)

[SaveTouchstoneFileFormatRealImage](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared SaveTouchstoneFileFormat As Integer
```

C#

```
public static int SaveTouchstoneFileFormat
```

Visual C++

```
public:
```

Visual C++

```
static int SaveTouchstoneFileFormat
```

JavaScript

```
CMT.Instruments.Attributes.saveTouchstoneFileFormat
```

Back to [Attributes](#)

SaveType

Description

Sets/Gets the type of the Analyzer or channel state saving by [SaveChannelToRegister](#) function.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[SaveTypeAll](#)

[SaveTypeState](#)

[SaveTypeStateAndCal](#)

[SaveTypeStateAndCalAndMem](#)

[SaveTypeStateAndTrace](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared SaveType As Integer
```

C#

```
public static int SaveType
```

Visual C++

```
public:  
static int SaveType
```

JavaScript

```
CMT.Instruments.Attributes.saveType
```

Back to [Attributes](#)

SelectedMarker

Description

Sets the active marker.

If the marker is not ON, this function will turn the marker ON. Turning ON a marker with the number from 1 to 15 will turn ON all the markers of smaller numbers. Turning ON the reference marker with number 16 does not turn ON the markers with the numbers from 1 to 15, but switches these markers to the relative measurement mode.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared SelectedMarker As Integer
```

C#

```
public static int SelectedMarker
```

Visual C++

```
public:  
static int SelectedMarker
```

JavaScript

CMT.Instruments.Attributes.selectedMarker

Back to [Attributes](#)

StimulusExtTriggerDelay

Description

Sets/Gets the response delay with respect to the external trigger signal. The value is expressed in seconds (sec).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusExtTriggerDelay As Integer
```

C#

```
public static int StimulusExtTriggerDelay
```

Visual C++

```
public:  
static int StimulusExtTriggerDelay
```

JavaScript

```
CMT.Instruments.Attributes.stimulusExtTriggerDelay
```

Back to [Attributes](#)

StimulusExtTriggerEvent

Description

Turns ON/OFF the point trigger feature for external trigger source.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[ExtTriggerEventOnSweep](#)

[ExtTriggerEventOnPoint](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusExtTriggerEvent As Integer
```

C#

```
public static int StimulusExtTriggerEvent
```

Visual C++

```
public:  
static int StimulusExtTriggerEvent
```

JavaScript

CMT.Instruments.Attributes.stimulusExtTriggerEvent

Back to [Attributes](#)

StimulusExtTriggerPolarity

Description

Sets/Gets out the polarity of the external trigger.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[ExtTriggerPolarityNegative](#)

[ExtTriggerPolarityPositive](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusExtTriggerPolarity As Integer
```

C#

```
public static int StimulusExtTriggerPolarity
```

Visual C++

```
public:  
static int StimulusExtTriggerPolarity
```

JavaScript

CMT.Instruments.Attributes.stimulusExtTriggerPolarity

Back to [Attributes](#)

StimulusExtTriggerPosition

Description

Sets/Gets the position of the external trigger.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[ExtTriggerPositionBsampling](#)

[ExtTriggerPositionBsetup](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusExtTriggerPosition As Integer
```

C#

```
public static int StimulusExtTriggerPosition
```

Visual C++

```
public:  
static int StimulusExtTriggerPosition
```

JavaScript

CMT.Instruments.Attributes.stimulusExtTriggerPosition

Back to [Attributes](#)

StimulusFrequencyCenter

Description

Sets/Gets the center value of the frequency sweep range. The value is expressed in hertz (Hz).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusFrequencyCenter As Integer
```

C#

```
public static int StimulusFrequencyCenter
```

Visual C++

```
public:  
static int StimulusFrequencyCenter
```

JavaScript

```
CMT.Instruments.Attributes.stimulusFrequencyCenter
```

Back to [Attributes](#)

StimulusFrequencyOffset

Description

Turns ON/OFF the frequency offset feature.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusFrequencyOffset As Integer
```

C#

```
public static int StimulusFrequencyOffset
```

Visual C++

```
public:  
static int StimulusFrequencyOffset
```

JavaScript

```
CMT.Instruments.Attributes.stimulusFrequencyOffset
```

Back to [Attributes](#)

StimulusFrequencySpan

Description

Sets/Gets the span value of the frequency sweep range. The value is expressed in hertz (Hz).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusFrequencySpan As Integer
```

C#

```
public static int StimulusFrequencySpan
```

Visual C++

```
public:  
static int StimulusFrequencySpan
```

JavaScript

```
CMT.Instruments.Attributes.stimulusFrequencySpan
```

Back to [Attributes](#)

StimulusFrequencyStart

Description

Sets/Gets the start value of the frequency sweep range. The value is expressed in hertz (Hz).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusFrequencyStart As Integer
```

C#

```
public static int StimulusFrequencyStart
```

Visual C++

```
public:  
static int StimulusFrequencyStart
```

JavaScript

```
CMT.Instruments.Attributes.stimulusFrequencyStart
```

Back to [Attributes](#)

StimulusFrequencyStop

Description

Sets/Gets the stop value of the frequency sweep range. The value is expressed in hertz (Hz).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusFrequencyStop As Integer
```

C#

```
public static int StimulusFrequencyStop
```

Visual C++

```
public:  
static int StimulusFrequencyStop
```

JavaScript

```
CMT.Instruments.Attributes.stimulusFrequencyStop
```

Back to [Attributes](#)

StimulusIfBandwidth

Description

Sets/Gets the bandwidth of the digital IF filter to be used in the measurement.

IF Bandwidth is in Hz. The list of valid IF Bandwidths is different depending on the analyzer model.

If an invalid number is specified, the analyzer will round up to the closest valid number.

The value is expressed in hertz (Hz).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusIfBandwidth As Integer
```

C#

```
public static int StimulusIfBandwidth
```

Visual C++

```
public:  
static int StimulusIfBandwidth
```

JavaScript

CMT.Instruments.Attributes.stimulusIfBandwidth

Back to [Attributes](#)

StimulusMaxFrequency

Description

Gets the upper limit of the measurement frequency. The value is expressed in hertz (Hz).

Used as attributeID with functions: [GetAttributeViReal64](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusMaxFrequency As Integer
```

C#

```
public static int StimulusMaxFrequency
```

Visual C++

```
public:  
static int StimulusMaxFrequency
```

JavaScript

```
CMT.Instruments.Attributes.stimulusMaxFrequency
```

Back to [Attributes](#)

StimulusMaxPoints

Description

Gets the maximum number of the measurement points.

Used as attributeID with functions: [GetAttributeViInt32](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusMaxPoints As Integer
```

C#

```
public static int StimulusMaxPoints
```

Visual C++

```
public:  
static int StimulusMaxPoints
```

JavaScript

```
CMT.Instruments.Attributes.stimulusMaxPoints
```

Back to [Attributes](#)

StimulusMinFrequency

Description

Gets the lower frequency of the measurement frequency. The value is expressed in hertz (Hz).

Used as attributeID with functions: [GetAttributeViReal64](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusMinFrequency As Integer
```

C#

```
public static int StimulusMinFrequency
```

Visual C++

```
public:  
static int StimulusMinFrequency
```

JavaScript

```
CMT.Instruments.Attributes.stimulusMinFrequency
```

Back to [Attributes](#)

StimulusOutputPower

Description

Sets/Gets the power level for the frequency sweep type. This value is expressed in decibels above 1 milliwatt (dBm).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusOutputPower As Integer
```

C#

```
public static int StimulusOutputPower
```

Visual C++

```
public:  
static int StimulusOutputPower
```

JavaScript

```
CMT.Instruments.Attributes.stimulusOutputPower
```

Back to [Attributes](#)

StimulusOutputPowerPort

Description

Sets/Gets the power level of port for the frequency sweep type when the port couple feature is set to OFF.

This value is expressed in decibels above 1 milliwatt (dBm).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index and Port index i.e. "Channel1:Port1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusOutputPowerPort As Integer
```

C#

```
public static int StimulusOutputPowerPort
```

Visual C++

```
public:  
static int StimulusOutputPowerPort
```

JavaScript

```
CMT.Instruments.Attributes.stimulusOutputPowerPort
```

Back to [Attributes](#)

StimulusOutputPowerPortCouple

Description

Turns ON/OFF the port power couple.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusOutputPowerPortCouple As Integer
```

C#

```
public static int StimulusOutputPowerPortCouple
```

Visual C++

```
public:  
static int StimulusOutputPowerPortCouple
```

JavaScript

```
CMT.Instruments.Attributes.stimulusOutputPowerPortCouple
```

Back to [Attributes](#)

StimulusOutputPowerRfout

Description

Turns ON/OFF the RF signal output. Measurements cannot be performed when the RF signal output is turned OFF.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusOutputPowerRfout As Integer
```

C#

```
public static int StimulusOutputPowerRfout
```

Visual C++

```
public:  
static int StimulusOutputPowerRfout
```

JavaScript

```
CMT.Instruments.Attributes.stimulusOutputPowerRfout
```

Back to [Attributes](#)

StimulusOutputPowerSlope

Description

Sets/Gets the power slope value for the frequency sweep. This value is expressed in decibels/gigahertz (dB/GHz).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusOutputPowerSlope As Integer
```

C#

```
public static int StimulusOutputPowerSlope
```

Visual C++

```
public:  
static int StimulusOutputPowerSlope
```

JavaScript

```
CMT.Instruments.Attributes.stimulusOutputPowerSlope
```

Back to [Attributes](#)

StimulusOutputPowerSlopeState

Description

Turns ON/OFF the power slope. The power slope is valid for the frequency sweep type: Linear, Logarithmic, Segment.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusOutputPowerSlopeState As Integer
```

C#

```
public static int StimulusOutputPowerSlopeState
```

Visual C++

```
public:  
static int StimulusOutputPowerSlopeState
```

JavaScript

```
CMT.Instruments.Attributes.stimulusOutputPowerSlopeState
```

Back to [Attributes](#)

StimulusPoints

Description

Sets/Gets the number of sweep points.

Used as attributeID with functions:

[SetAttributeVInt32](#)

[GetAttributeVInt32](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusPoints As Integer
```

C#

```
public static int StimulusPoints
```

Visual C++

```
public:  
static int StimulusPoints
```

JavaScript

```
CMT.Instruments.Attributes.stimulusPoints
```

Back to [Attributes](#)

StimulusPowerCenter

Description

Sets/Gets the center value of the power sweep type. The value is expressed in decibels above 1 milliwatt (dBm).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusPowerCenter As Integer
```

C#

```
public static int StimulusPowerCenter
```

Visual C++

```
public:  
static int StimulusPowerCenter
```

JavaScript

```
CMT.Instruments.Attributes.stimulusPowerCenter
```

Back to [Attributes](#)

StimulusPowerCwFrequency

Description

Sets/Gets the fixed frequency value when the power sweep type is selected. The value is expressed in hertz (Hz).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusPowerCwFrequency As Integer
```

C#

```
public static int StimulusPowerCwFrequency
```

Visual C++

```
public:  
static int StimulusPowerCwFrequency
```

JavaScript

```
CMT.Instruments.Attributes.stimulusPowerCwFrequency
```

Back to [Attributes](#)

StimulusPowerSpan

Description

Sets/Gets the power span when the power sweep type is active. The value is expressed in decibels above 1 milliwatt (dBm).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusPowerSpan As Integer
```

C#

```
public static int StimulusPowerSpan
```

Visual C++

```
public:  
static int StimulusPowerSpan
```

JavaScript

```
CMT.Instruments.Attributes.stimulusPowerSpan
```

Back to [Attributes](#)

StimulusPowerStart

Description

Sets/Gets the power sweep start value when the power sweep type is active. The value is expressed in decibels above 1 milliwatt (dBm).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusPowerStart As Integer
```

C#

```
public static int StimulusPowerStart
```

Visual C++

```
public:  
static int StimulusPowerStart
```

JavaScript

```
CMT.Instruments.Attributes.stimulusPowerStart
```

Back to [Attributes](#)

StimulusPowerStop

Description

Sets/Gets the power sweep stop value when the power sweep type is active. The value is expressed in decibels above 1 milliwatt (dBm).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusPowerStop As Integer
```

C#

```
public static int StimulusPowerStop
```

Visual C++

```
public:  
static int StimulusPowerStop
```

JavaScript

```
CMT.Instruments.Attributes.stimulusPowerStop
```

Back to [Attributes](#)

StimulusSegmentDisplayOrder

Description

Sets/Gets the display method of the graph horizontal axis for the segment sweep.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[SegmentFrequencyOrder](#)

[SegmentIndexOrder](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusSegmentDisplayOrder As Integer
```

C#

```
public static int StimulusSegmentDisplayOrder
```

Visual C++

```
public:  
static int StimulusSegmentDisplayOrder
```

JavaScript

CMT.Instruments.Attributes.stimulusSegmentDisplayOrder

Back to [Attributes](#)

StimulusSweepMeasureDelay

Description

Sets/Gets the delay before measurement in each measurement point. The value is expressed in seconds (sec).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusSweepMeasureDelay As Integer
```

C#

```
public static int StimulusSweepMeasureDelay
```

Visual C++

```
public:  
static int StimulusSweepMeasureDelay
```

JavaScript

```
CMT.Instruments.Attributes.stimulusSweepMeasureDelay
```

Back to [Attributes](#)

StimulusSweepType

Description

Sets/Gets the sweep type of channel.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[SweepTypeLinFrequency](#)

[SweepTypeLogFrequency](#)

[SweepTypeSegment](#)

[SweepTypePower](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusSweepType As Integer
```

C#

```
public static int StimulusSweepType
```

Visual C++

```
public:  
static int StimulusSweepType
```

JavaScript

CMT.Instruments.Attributes.stimulusSweepType

Back to [Attributes](#)

StimulusTriggerMode

Description

Sets/Gets the trigger mode.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[TriggerModeHold](#)

[TriggerModeSingle](#)

[TriggerModeContinuous](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusTriggerMode As Integer
```

C#

```
public static int StimulusTriggerMode
```

Visual C++

```
public:  
static int StimulusTriggerMode
```

JavaScript

CMT.Instruments.Attributes.stimulusTriggerMode

Back to [Attributes](#)

StimulusTriggerOutput

Description

Turns ON/OFF the trigger output.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusTriggerOutput As Integer
```

C#

```
public static int StimulusTriggerOutput
```

Visual C++

```
public:  
static int StimulusTriggerOutput
```

JavaScript

```
CMT.Instruments.Attributes.stimulusTriggerOutput
```

Back to [Attributes](#)

StimulusTriggerOutputFunction

Description

Sets/Gets the trigger output function.

The trigger output outputs various waveforms depending on the setting of the Output Trigger Function.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[TriggerOutputFunctionAsampling](#)

[TriggerOutputFunctionBsampling](#)

[TriggerOutputFunctionBsetup](#)

[TriggerOutputFunctionMeasurement](#)

[TriggerOutputFunctionReadyForTrigger](#)

[TriggerOutputFunctionSweepEnd](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusTriggerOutputFunction As Integer
```

C#

```
public static int StimulusTriggerOutputFunction
```

Visual C++

```
public:  
static int StimulusTriggerOutputFunction
```

JavaScript

```
CMT.Instruments.Attributes.stimulusTriggerOutputFunction
```

Back to [Attributes](#)

StimulusTriggerOutputPolarity

Description

Sets/Gets the polarity of the trigger output.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[TriggerOutputPolarityNegative](#)

[TriggerOutputPolarityPositive](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusTriggerOutputPolarity As Integer
```

C#

```
public static int StimulusTriggerOutputPolarity
```

Visual C++

```
public:  
static int StimulusTriggerOutputPolarity
```


JavaScript

CMT.Instruments.Attributes.stimulusTriggerOutputPolarity

Back to [Attributes](#)

StimulusTriggerScope

Description

Sets/Gets the trigger scope.

The trigger scope determines the response on the trigger signal arrival: either starts a sweep of all waiting repCapIDs in turn or starts a sweep in the active channel only.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[TriggerScopeActiveChannel](#)

[TriggerScopeAllChannel](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusTriggerScope As Integer
```

C#

```
public static int StimulusTriggerScope
```

Visual C++

```
public:  
static int StimulusTriggerScope
```

JavaScript

CMT.Instruments.Attributes.stimulusTriggerScope

Back to [Attributes](#)

StimulusTriggerSource

Description

Sets/Gets the trigger source.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[TriggerSourceInternal](#)

[TriggerSourceExternal](#)

[TriggerSourceManual](#)

[TriggerSourceBus](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusTriggerSource As Integer
```

C#

```
public static int StimulusTriggerSource
```

Visual C++

```
public:  
static int StimulusTriggerSource
```

JavaScript

CMT.Instruments.Attributes.stimulusTriggerSource

Back to [Attributes](#)

StimulusTriggerState

Description

Gets the the current state of the analyzer.

Used as attributeID with functions: [GetAttributeViInt32](#)

Field Value

[StimulusTriggerStateHold](#)

[StimulusTriggerStateMeasure](#)

[StimulusTriggerStateWait](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusTriggerState As Integer
```

C#

```
public static int StimulusTriggerState
```

Visual C++

```
public:  
static int StimulusTriggerState
```

JavaScript

```
CMT.Instruments.Attributes.stimulusTriggerState
```

Back to [Attributes](#)

SystemCorrection

Description

Turns ON/OFF the system correction.

The system correction is the factory full 1-port calibration performed at the port connectors.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared SystemCorrection As Integer
```

C#

```
public static int SystemCorrection
```

Visual C++

```
public:  
static int SystemCorrection
```

JavaScript

```
CMT.Instruments.Attributes.systemCorrection
```

Back to [Attributes](#)

SystemReadWriteLock

Description

Sets the Analyzer to the local operation mode or remote operation mode, when all the keys on the front panel, mouse and the touch screen are active.

Used as attributeID with function [SetAttributeViBoolean](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared SystemReadWriteLock As Integer
```

C#

```
public static int SystemReadWriteLock
```

Visual C++

```
public:  
static int SystemReadWriteLock
```

JavaScript

```
CMT.Instruments.Attributes.systemReadWriteLock
```

Back to [Attributes](#)

SystemTimeoutMilliseconds

Description

I/O timeout value in milliseconds. This value is expressed in milliseconds (ms).

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared SystemTimeoutMilliseconds As Integer
```

C#

```
public static int SystemTimeoutMilliseconds
```

Visual C++

```
public:  
static int SystemTimeoutMilliseconds
```

JavaScript

```
CMT.Instruments.Attributes.systemTimeoutMilliseconds
```

Back to [Attributes](#)

ThruAdditionCutoffFreq

Description

Sets/Gets the value of the cutoff frequency of the waveguide thru in the thru addition function.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ThruAdditionCutoffFreq As Integer
```

C#

```
public static int ThruAdditionCutoffFreq
```

Visual C++

```
public:  
static int ThruAdditionCutoffFreq
```

JavaScript

```
CMT.Instruments.Attributes.thruAdditionCutoffFreq
```

Back to [Attributes](#)

ThruAdditionDelay

Description

Sets/Gets the approximate delay value of an unknown thru in the thru addition function.

This value is used to eliminate the uncertainty of ± 180 deg when calculating the phase response of the thru. If this value is set to zero, the analyzer uses an algorithm to automatically determine the delay of the thru.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ThruAdditionDelay As Integer
```

C#

```
public static int ThruAdditionDelay
```

Visual C++

```
public:  
static int ThruAdditionDelay
```

JavaScript

```
CMT.Instruments.Attributes.thruAdditionDelay
```

Back to [Attributes](#)

ThruAdditionLength

Description

Sets/Gets the approximate value of the mechanical length of an unknown thru in the thru addition function.

This value is used to eliminate the uncertainty of ± 180 deg when calculating the phase response of the thru. If this value is set to zero, the analyzer uses an algorithm to automatically determine the delay of the thru.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ThruAdditionLength As Integer
```

C#

```
public static int ThruAdditionLength
```

Visual C++

```
public:  
static int ThruAdditionLength
```

JavaScript

```
CMT.Instruments.Attributes.thruAdditionLength
```

Back to [Attributes](#)

ThruAdditionMedia

Description

Sets/Gets the media of the thru in the thru addition function.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[ThruAdditionMediaCoaxial](#)

[ThruAdditionMediaWaveguide](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ThruAdditionMedia As Integer
```

C#

```
public static int ThruAdditionMedia
```

Visual C++

```
public:  
static int ThruAdditionMedia
```


JavaScript

CMT.Instruments.Attributes.thruAdditionMedia

Back to [Attributes](#)

ThruAdditionPermittivity

Description

Sets/Gets the value of the permittivity of the thru media in the thru addition function.

This parameter is used to calculate the adapter delay when the thru length is being set; therefore this parameter must be set before setting of the thru length.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ThruAdditionPermittivity As Integer
```

C#

```
public static int ThruAdditionPermittivity
```

Visual C++

```
public:  
static int ThruAdditionPermittivity
```

JavaScript

```
CMT.Instruments.Attributes.thruAdditionPermittivity
```

Back to [Attributes](#)

ThruAdditionUnit

Description

Sets/Gets the display units of the thru delay (length) in the thru addition function.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[ThruAdditionUnitMeters](#)

[ThruAdditionUnitSeconds](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ThruAdditionUnit As Integer
```

C#

```
public static int ThruAdditionUnit
```

Visual C++

```
public:  
static int ThruAdditionUnit
```

JavaScript

CMT.Instruments.Attributes.thruAdditionUnit

Back to [Attributes](#)

TimeDomainCableCorrection

Description

Turns ON/OFF the cable correction when the time domain transformation function is turned ON.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainCableCorrection As Integer
```

C#

```
public static int TimeDomainCableCorrection
```

Visual C++

```
public:  
static int TimeDomainCableCorrection
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainCableCorrection
```

Back to [Attributes](#)

TimeDomainCableFrequency

Description

Sets/Gets the frequency value at which the cable loss is specified for the cable correction function, when the time domain transformation function is turned ON. The value is expressed in Hertz (Hz).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainCableFrequency As Integer
```

C#

```
public static int TimeDomainCableFrequency
```

Visual C++

```
public:  
static int TimeDomainCableFrequency
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainCableFrequency
```

Back to [Attributes](#)

TimeDomainCableLoss

Description

Sets/Gets the cable loss value for the cable correction function, when the time domain transformation function is turned ON.

The value is expressed in decibell/meter (dB/m).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainCableLoss As Integer
```

C#

```
public static int TimeDomainCableLoss
```

Visual C++

```
public:  
static int TimeDomainCableLoss
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainCableLoss
```

Back to [Attributes](#)

TimeDomainCableVelocityFactor

Description

Sets/Gets the cable relative wave speed velocity for the cable correction function, when the time domain transformation function is turned ON.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index i.e. "Channel1" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainCableVelocityFactor As Integer
```

C#

```
public static int TimeDomainCableVelocityFactor
```

Visual C++

```
public:  
static int TimeDomainCableVelocityFactor
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainCableVelocityFactor
```

Back to [Attributes](#)

TimeDomainCenter

Description

Sets/Gets the time domain center value, when the time domain transformation function is turned ON. The value is expressed in second (sec).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainCenter As Integer
```

C#

```
public static int TimeDomainCenter
```

Visual C++

```
public:  
static int TimeDomainCenter
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainCenter
```

Back to [Attributes](#)

TimeDomainGating

Description

Turns ON/OFF the gating function.

Used as attributeID with functions:

[SetAttributeViBoolean](#)

[GetAttributeViBoolean](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainGating As Integer
```

C#

```
public static int TimeDomainGating
```

Visual C++

```
public:  
static int TimeDomainGating
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainGating
```

Back to [Attributes](#)

TimeDomainGatingCenter

Description

Sets/Gets the gate center value of the gating function. The value is expressed in seconds (sec).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainGatingCenter As Integer
```

C#

```
public static int TimeDomainGatingCenter
```

Visual C++

```
public:  
static int TimeDomainGatingCenter
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainGatingCenter
```

Back to [Attributes](#)

TimeDomainGatingShape

Description

Sets/Gets the gate shape of the gating function.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[TimeDomainGatingShapeMaximum](#)

[TimeDomainGatingShapeMinimum](#)

[TimeDomainGatingShapeNormal](#)

[TimeDomainGatingShapeWide](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainGatingShape As Integer
```

C#

```
public static int TimeDomainGatingShape
```

Visual C++

```
public:
```

Visual C++

```
static int TimeDomainGatingShape
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainGatingShape
```

Back to [Attributes](#)

TimeDomainGatingSpan

Description

Sets/Gets the gate span value of the gating function. The value is expressed in seconds (sec).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainGatingSpan As Integer
```

C#

```
public static int TimeDomainGatingSpan
```

Visual C++

```
public:  
static int TimeDomainGatingSpan
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainGatingSpan
```

Back to [Attributes](#)

TimeDomainGatingStart

Description

Sets/Gets the gate start value of the gating function. The value is expressed in seconds (sec).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainGatingStart As Integer
```

C#

```
public static int TimeDomainGatingStart
```

Visual C++

```
public:  
static int TimeDomainGatingStart
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainGatingStart
```

Back to [Attributes](#)

TimeDomainGatingStop

Description

Sets/Gets the gate stop value of the gating function. The value is expressed in seconds (sec).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainGatingStop As Integer
```

C#

```
public static int TimeDomainGatingStop
```

Visual C++

```
public:  
static int TimeDomainGatingStop
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainGatingStop
```

Back to [Attributes](#)

TimeDomainGatingType

Description

Sets/Gets the gate type of the gating function.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[TimeDomainGatingBandpass](#)

[TimeDomainGatingNotch](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainGatingType As Integer
```

C#

```
public static int TimeDomainGatingType
```

Visual C++

```
public:  
static int TimeDomainGatingType
```

JavaScript

CMT.Instruments.Attributes.timeDomainGatingType

Back to [Attributes](#)

TimeDomainImpulseWidth

Description

Sets/Gets the impulse width (time domain transformation resolution), coupled with the Kaiser–Bessel window shape Beta parameter.

The impulse width setting changes the Beta parameter, and setting of Beta parameter changes the impulse width.

The value is expressed in seconds (sec).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainImpulseWidth As Integer
```

C#

```
public static int TimeDomainImpulseWidth
```

Visual C++

```
public:  
static int TimeDomainImpulseWidth
```

JavaScript

CMT.Instruments.Attributes.timeDomainImpulseWidth

Back to [Attributes](#)

TimeDomainKaiserBeta

Description

Sets/Gets the Beta parameter, which controls the Kaiser–Bessel window shape, when performing time domain transformation.

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainKaiserBeta As Integer
```

C#

```
public static int TimeDomainKaiserBeta
```

Visual C++

```
public:  
static int TimeDomainKaiserBeta
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainKaiserBeta
```

Back to [Attributes](#)

TimeDomainSpan

Description

Sets/Gets the time domain span value, when the time domain transformation function is turned ON. The value is expressed in second (sec).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainSpan As Integer
```

C#

```
public static int TimeDomainSpan
```

Visual C++

```
public:  
static int TimeDomainSpan
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainSpan
```

Back to [Attributes](#)

TimeDomainStart

Description

Sets/Gets the time domain start value, when the time domain transformation function is turned ON. The value is expressed in second (sec).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainStart As Integer
```

C#

```
public static int TimeDomainStart
```

Visual C++

```
public:  
static int TimeDomainStart
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainStart
```

Back to [Attributes](#)

TimeDomainStop

Description

Sets/Gets the time domain stop value, when the time domain transformation function is turned ON.

The value is expressed in second (sec).

Used as attributeID with functions:

[SetAttributeViReal64](#)

[GetAttributeViReal64](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainStop As Integer
```

C#

```
public static int TimeDomainStop
```

Visual C++

```
public:  
static int TimeDomainStop
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainStop
```


Back to [Attributes](#)

TimeDomainTransformType

Description

Sets/Gets the stimulus type for the time domain transformation function.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[TimeDomainTransformTypeBandpass](#)

[TimeDomainTransformTypeLowpassImpulse](#)

[TimeDomainTransformTypeLowpassStep](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n-th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainTransformType As Integer
```

C#

```
public static int TimeDomainTransformType
```

Visual C++

```
public:  
static int TimeDomainTransformType
```

JavaScript

CMT.Instruments.Attributes.timeDomainTransformType

Back to [Attributes](#)

TimeDomainWindowShape

Description

Sets/Gets the Kaiser–Bessel window shape, when performing time domain transformation.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

Field Value

[TimeDomainWindowShapeArbitrary](#)

[TimeDomainWindowShapeMaximum](#)

[TimeDomainWindowShapeMinimum](#)

[TimeDomainWindowShapeNormal](#)

The repeated capability identifier:

The physical names are supported along with the corresponding Channel index for the active trace of channel i.e. "Channel1" and so on. The physical names are supported along with the corresponding Channel index and Measurement index for the n–th trace of channel i.e. "Channel1:Measurement3" and so on.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainWindowShape As Integer
```

C#

```
public static int TimeDomainWindowShape
```

Visual C++

```
public:  
static int TimeDomainWindowShape
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainWindowShape
```

Back to [Attributes](#)

VerificationInterval

Description

Sets/Gets the interval between Instrument Performance Verifications.

One year (365 days) is recommended.

This value is expressed in days.

Used as attributeID with functions:

[SetAttributeViInt32](#)

[GetAttributeViInt32](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared VerificationInterval As Integer
```

C#

```
public static int VerificationInterval
```

Visual C++

```
public:  
static int VerificationInterval
```

JavaScript

```
CMT.Instruments.Attributes.verificationInterval
```

Back to [Attributes](#)

VerificationLastDate

Description

Sets/Gets the date of the last Instrument Performance Verification.

The date format: "YYYY,MM,DD" (YYYY - year, MM - month, DD - day).

Used as attributeID with functions:

[SetAttributeViString](#)

[GetAttributeViString](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared VerificationLastDate As Integer
```

C#

```
public static int VerificationLastDate
```

Visual C++

```
public:  
static int VerificationLastDate
```

JavaScript

```
CMT.Instruments.Attributes.verificationLastDate
```

Back to [Attributes](#)

VerificationNextDate

Description

Gets the date of the next Instrument Performance Verification.

The date format: "YYYY,MM,DD" (YYYY - year, MM - month, DD - day).

Used as attributeID with functions:

[SetAttributeViString](#)

[GetAttributeViString](#)

The repeated capability identifier:

Must be VI_NULL or an empty string. This attribute is not defined on a repeated capability.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared VerificationNextDate As Integer
```

C#

```
public static int VerificationNextDate
```

Visual C++















```
public:  
static int VerificationNextDate
```












JavaScript



```
CMT.Instruments.Attributes.verificationNextDate
```

Back to [Attributes](#)







class Constants












	Name	Description
	<u>AdapterRemovalMediaCoaxial</u>	The adapter media - Specifies the coaxial adapter
	<u>AdapterRemovalMediaWaveguide</u>	The adapter media - Specifies the waveguide adapter
	<u>AdapterRemovalUnitMeters</u>	The display units of the adapter delay - Selects meters
	<u>AdapterRemovalUnitSeconds</u>	The display units of the adapter delay - Selects seconds
	<u>AnalysisLimitLineMax</u>	The limit line type - Maximum
	<u>AnalysisLimitLineMin</u>	The limit line type - Minimum
	<u>AnalysisLimitLineOff</u>	The limit line type - Limit Off
	<u>AnalysisLimitLineSingle</u>	Te limit line type - Single Line
	<u>AnalysisRippleLimitOff</u>	Ripple value display OFF
	<u>AnalysisRippleLimitOn</u>	Ripple value display ON
	<u>AnalysisRippleValueAbsolute</u>	The display type of the ripple value - Absolute value
	<u>AnalysisRippleValueMargin</u>	The display type of the ripple value - Margin (difference between the ripple limit and the absolute value)
	<u>AnalysisRippleValueOff</u>	The display type of the ripple value - Ripple value display OFF
	<u>Autocal1port</u>	The type AutoCal Module calibration - Full 1-port calibration








	Name	Description
	Autocal2port	The type AutoCal Module calibration - Full 2-port calibration
	Autocal3port	The type AutoCal Module calibration - Full 3-port calibration
	Autocal4port	The type AutoCal Module calibration - Full 4-port calibration
	AutocalCharacterizationFactory	The characterization number used when executing AutoCal - Factory characterization
	AutocalCharacterizationUser1	The characterization number used when executing AutoCal - User characterization 1
	AutocalCharacterizationUser2	The characterization number used when executing AutoCal - User characterization 2
	AutocalCharacterizationUser3	The characterization number used when executing AutoCal - User characterization 3
	AutocalOnePath2port	The type AutoCal Module calibration - 1 Path 2-port calibration
	AutocalOrientationPortA	Port A of AutoCal Module which is connected to a specified port of Network Analyzer
	AutocalOrientationPortB	Port B of AutoCal Module which is connected to a specified port of Network Analyzer
	AutocalOrientationPortC	Port C of AutoCal Module which is connected to a specified port of Network Analyzer

	Name	Description
	<u>AutocalOrientationPortD</u>	Port D of AutoCal Module which is connected to a specified port of Network Analyzer
	<u>AutoPortExtensionActiveMarker</u>	Uses the frequency of the active marker for calculation to the Port Extension
	<u>AutoPortExtensionCurrentSpan</u>	Uses current frequency span for calculation to the Port Extension
	<u>AutoPortExtensionUserSpan</u>	Uses arbitrary frequency range set by user for calculation to the Port Extension
	<u>CalibrationActionIsolation</u>	The type of calibration action - Isolation calibration
	<u>CalibrationActionLoad</u>	The type of calibration action - LOAD calibration
	<u>CalibrationActionOpen</u>	The type of calibration action - OPEN calibration
	<u>CalibrationActionPowerCal</u>	The type of calibration action - The power calibration
	<u>CalibrationActionReferenceReceiverCal</u>	The type of calibration action - The reference receiver calibration
	<u>CalibrationActionShort</u>	The type of calibration action - SHOT calibration
	<u>CalibrationActionTestReceiverCal</u>	The type of calibration action - The test receiver calibration
	<u>CalibrationActionThru</u>	The type of calibration action - THRU calibration













	Name	Description
	CalibrationTriggerSourceInternal	The internal trigger source for calibration
	CalibrationTriggerSourceSystem	The system trigger source for calibration
	CalibrationType1path2port	The calibration type for the calculation of the calibration coefficients - One-path 2-port
	CalibrationType1portSol	The calibration type for the calculation of the calibration coefficients - Full 1-port
	CalibrationType2portSolt	The calibration type for the calculation of the calibration coefficients - Full 2-port
	CalibrationType2portTrl	The calibration type for the calculation of the calibration coefficients - TRL 2-port
	CalibrationType3portSolt	The calibration type for the calculation of the calibration coefficients - Full 3-port
	CalibrationType3portTrl	The calibration type for the calculation of the calibration coefficients - TRL 3-port
	CalibrationType4portSolt	The calibration type for the calculation of the calibration coefficients - Full 4-port
	CalibrationType4portTrl	The calibration type for the calculation of the calibration coefficients - TRL 4-port
	CalibrationTypeAdapterRemoval	The calibration type for the calculation of the calibration coefficients - Adapter Removal





	Name	Description
	<u>CalibrationTypeNotDefined</u>	The calibration type for the calculation of the calibration coefficients - Not defined
	<u>CalibrationTypeResponseOpen</u>	The calibration type for the calculation of the calibration coefficients - OPEN response
	<u>CalibrationTypeResponseShort</u>	The calibration type for the calculation of the calibration coefficients - SHORT response
	<u>CalibrationTypeResponseThru</u>	The calibration type for the calculation of the calibration coefficients - THRU response
	<u>CalkitStandardDataBased</u>	The type of calibration standard - Data Based
	<u>CalkitStandardLoad</u>	The type of calibration standard - Load
	<u>CalkitStandardNone</u>	The type of calibration standard - Not defined
	<u>CalkitStandardOpen</u>	The type of calibration standard - Open
	<u>CalkitStandardShort</u>	The type of calibration standard - Short
	<u>CalkitStandardSlidingLoad</u>	The type of calibration standard - Sliding Load
	<u>CalkitStandardThruDelay</u>	The type of calibration standard - Thru
	<u>CalkitStandardUnknThru</u>	The type of calibration standard - Unknown Thru

	Name	Description
	<u>ChannelStateRegisterA</u>	Saves the Analyzer state of the active channel into memory register A
	<u>ChannelStateRegisterB</u>	Saves the Analyzer state of the active channel into memory register B.
	<u>ChannelStateRegisterC</u>	Saves the Analyzer state of the active channel into memory register C.
	<u>ChannelStateRegisterD</u>	Saves the Analyzer state of the active channel into memory register D.
	<u>ConversionFunctionConjugation</u>	S-parameter conversion function type - S-parameter conjugate
	<u>ConversionFunctionSInverse</u>	S-parameter conversion function type - Inverse S-parameter
	<u>ConversionFunctionYReflection</u>	S-parameter conversion function type - Reflection equivalent admittance
	<u>ConversionFunctionYTransmission</u>	S-parameter conversion function type - Transmission equivalent admittance
	<u>ConversionFunctionYTransShunt</u>	S-parameter conversion function type - Shunt equivalent admittance
	<u>ConversionFunctionZReflection</u>	S-parameter conversion function type - Reflection equivalent impedance
	<u>ConversionFunctionZTransmission</u>	S-parameter conversion function type - Transmission equivalent impedance











	Name	Description
	ConversionFunctionZTransShunt	S-parameter conversion function type - Shunt equivalent impedance
	CorrectionStateCorrection	Correction status of the error correction
	CorrectionStateExtrapolation	Extrapolation status of the error correction
	CorrectionStateInterpolation	Interpolation status of the error correction
	CorrectionStateNone	None status of the error correction
	DisplayTraceData	Only data trace is displayed
	DisplayTraceDataAndMemory	Data trace and memory trace are displayed
	DisplayTraceMemory	Only memory trace is displayed
	DisplayTraceOff	Both traces are not displayed
	ExtTriggerEventOnPoint	The external trigger response is the single point
	ExtTriggerEventOnSweep	The external trigger response is the entire sweep
	ExtTriggerPolarityNegative	The polarity of the external trigger - Negative edge
	ExtTriggerPolarityPositive	The polarity of the external trigger - Positive edge
	ExtTriggerPositionBsampling	The position of the external trigger - Before sampling
	ExtTriggerPositionBsetup	The position of the external trigger - Before frequency setup













	Name	Description
	FrequencyOffsetTypePortPort	The frequency offset type - Port1/Port2 offset
	FrequencyOffsetTypeSourceReceiver	The frequency offset type - Source/Receivers offset
	FunctionAllPeaks	The type of analysis - Search for all the peaks
	FunctionAllTarget	The type of analysis - Search for all targets
	FunctionMaximum	The type of analysis - Maximum value
	FunctionMean	The type of analysis - Mean value
	FunctionMinimum	The type of analysis - Minimum value
	FunctionPeak	The type of analysis - Search for peak
	FunctionPeakPolarityBoth	The polarity when performing the peak search - Both: positive peaks and negative peaks
	FunctionPeakPolarityNegative	The polarity when performing the peak search - Negative peaks
	FunctionPeakPolarityPositive	The polarity when performing the peak search - Positive peaks
	FunctionPeakToPeak	The type of analysis - Peak-to-peak (difference between the maximum value and the minimum value)
	FunctionStandardDeviation	The type of analysis - Standard deviation


	Name	Description
	FunctionTransitionTypeBoth	The transition type when performing the search for the trace - Positive and Negative
	FunctionTransitionTypeNegative	The transition type when performing the search for the trace - Negative
	FunctionTransitionTypePositive	The transition type when performing the search for the trace - Positive
	MarkerMathBandwidthSearchBandpass	The type of the bandwidth search function - Bandpass
	MarkerMathBandwidthSearchNotch	The type of the bandwidth search function - Notch
	MarkerMathBandwidthSearchRefMarker	Bandwidth search relative to the reference marker
	MarkerMathBandwidthSearchRefMaximum	Bandwidth search relative to the absolute maximum of the trace
	MarkerMathBandwidthSearchRefMinimum	Bandwidth search relative to the absolute minimum of the trace
	MarkerPropertiesAlignHorizontal	The alignment mode of the marker display position - Horizontal
	MarkerPropertiesAlignOff	The alignment mode of the marker display position - No alignment
	MarkerPropertiesAlignVertical	The alignment mode of the marker display position - Vertical
	MarkerSearchPeakPolarityBoth	The peak polarity, when the marker search for peak - Both positive polarity and negative polarity










	Name	Description
	<u>MarkerSearchPeakPolarityNegative</u>	The peak polarity, when the marker search for peak - Negative polarity
	<u>MarkerSearchPeakPolarityPositive</u>	The peak polarity, when the marker search for peak - Positive polarity
	<u>MarkerSearchTargetTransitionBoth</u>	The type of the target transition, when the marker search for transition - Both positive target transition and negative target transition
	<u>MarkerSearchTargetTransitionNegative</u>	The type of the target transition, when the marker search for transition - Negative target transition
	<u>MarkerSearchTargetTransitionPositive</u>	The type of the target transition, when the marker search for transition - Positive target transition
	<u>MarkerSearchTypeMaximum</u>	The type of the marker search - Maximum
	<u>MarkerSearchTypeMinimum</u>	The type of the marker search - Minimum
	<u>MarkerSearchTypePeak</u>	The type of the marker search - Search Peak
	<u>MarkerSearchTypePeakLeft</u>	The type of the marker search - Search Peak Left
	<u>MarkerSearchTypePeakRight</u>	The type of the marker search - Search Peak Right
	<u>MarkerSearchTypeTarget</u>	The type of the marker search - Search Target














	Name	Description
	<u>MarkerSearchTypeTargetLeft</u>	The type of the marker search - Search Target Left
	<u>MarkerSearchTypeTargetRight</u>	The type of the marker search - Search Target Right
	<u>MeasurementFormatExpandPhase</u>	Measurement Format - Expanded phase
	<u>MeasurementFormatGroupDelay</u>	Measurement Format - Group delay time
	<u>MeasurementFormatImag</u>	Measurement Format - Imaginary part
	<u>MeasurementFormatLinMag</u>	Measurement Format - Linear magnitude
	<u>MeasurementFormatLogMag</u>	Measurement Format - Logarithmic magnitude
	<u>MeasurementFormatPhase</u>	Measurement Format - Phase
	<u>MeasurementFormatPolarLin</u>	Measurement Format - Polar format (Lin)
	<u>MeasurementFormatPolarLog</u>	Measurement Format - Polar format (Log)
	<u>MeasurementFormatPolarReallmage</u>	Measurement Format - Polar format (Real/Imag)
	<u>MeasurementFormatReal</u>	Measurement Format - Real part
	<u>MeasurementFormatSmithAdmittance</u>	Measurement Format - Smith chart format (G + jB)
	<u>MeasurementFormatSmithComplex</u>	Measurement Format - Smith chart format (R + jX)

	Name	Description
	MeasurementFormatSmithLin	Measurement Format - Smith chart format (Lin)
	MeasurementFormatSmithLog	Measurement Format - Smith chart format (Log)
	MeasurementFormatSmithRealIma ge	Measurement Format - Smith chart format (Real/Imag)
	MeasurementFormatSwr	Measurement Format - Voltage standing wave ratio
	MeasurementTypeAbsoluteR	The measurement parameter of the trace - Reference receiver: R1, R2, R3, R4
	MeasurementTypeAbsoluteT	The measurement parameter of the trace - Test receiver: A, B, C, D or T1, T2, T3, T4
	MeasurementTypeDcVoltage	The measurement parameter of the trace - DC Voltage: AUX1, AUX2 or V1, V2
	MeasurementTypeSParam	The measurement parameter of the trace - S parameter: S11, S12, S13, S14, S21, S22, S23, S24, S31, S32, S33, S34, S41, S42, S43, S44
	PowerSensorNrpzt	The power sensor type to be used in a source power calibration - Rohde_Schwarz NRPxT series Sensors
	PowerSensorNrpz	The power sensor type to be used in a source power calibration - Rohde_Schwarz NRP-Z series Sensors

	Name	Description
	PowerSensorNrvs	The power sensor type to be used in a source power calibration - Rohde_Schwarz NRVS power meter
	PowerSensorU200x	The power sensor type to be used in a source power calibration - Keysight U200x series Sensors
	PowerSensorU848x	The power sensor type to be used in a source power calibration - Keysight U848x series Sensors
	PrintBlackAndWhite	The color chart for the image printout - Black and white printout
	PrintColor	The color chart for the image printout - Color printout
	PrintGrayScale	The color chart for the image printout - Grayscale printout
	ReferenceFrequencySourceExternal	External source of the reference frequency of 10 MHz
	ReferenceFrequencySourceInternal	Internal source of the reference frequency of 10 MHz
	ReferenceRouteFront	The route of the external 10 MHz reference frequency - Front panel
	ReferenceRouteRear	The route of the external 10 MHz reference frequency - Rear panel
	SaveTouchstoneFileColumnSeparatorSpace	The Touchstone file separator symbol - Space symbol (0x20)
	SaveTouchstoneFileColumnSeparatorTab	The Touchstone file separator symbol - Tab symbol (0x09)












	Name	Description
	<u>SaveTouchstoneFileFormatDbAngle</u>	The data format for the S-parameter saving - Db / Angle
	<u>SaveTouchstoneFileFormatMagnitudeAngle</u>	The data format for the S-parameter saving - Magnitude / Angle
	<u>SaveTouchstoneFileFormatRealImage</u>	The data format for the S-parameter saving - Real / Image
	<u>SaveTouchstoneFileS1p</u>	The Touchstone file type when saving S-parameters - S1P
	<u>SaveTouchstoneFileS2p</u>	The Touchstone file type when saving S-parameters - S2P
	<u>SaveTouchstoneFileS3p</u>	The Touchstone file type when saving S-parameters - S3P
	<u>SaveTouchstoneFileS4p</u>	The Touchstone file type when saving S-parameters - S4P
	<u>SaveTypeAll</u>	The type of the Analyzer or channel state - Measurement conditions, calibration, data and memory
	<u>SaveTypeState</u>	The type of the Analyzer or channel state - Measurement conditions
	<u>SaveTypeStateAndCal</u>	The type of the Analyzer or channel state - Measurement conditions and calibration
	<u>SaveTypeStateAndCalAndMem</u>	The type of the Analyzer or channel state - Measurement conditions, calibration and memory
	<u>SaveTypeStateAndTrace</u>	The type of the Analyzer or channel state - Measurement conditions and data






	Name	Description
	<u>SegmentFrequencyOrder</u>	Frequency base (linear frequency axis)
	<u>SegmentIndexOrder</u>	Order base (linear axis of the point number)
	<u>StimulusTriggerStateHold</u>	The current state of the analyzer - Hold
	<u>StimulusTriggerStateMeasure</u>	The current state of the analyzer - Measure (sweep in progress)
	<u>StimulusTriggerStateWait</u>	The current state of the analyzer - Waiting for trigger
	<u>SweepTypeLinFrequency</u>	Linear frequency sweep
	<u>SweepTypeLogFrequency</u>	Logarithmic frequency sweep
	<u>SweepTypePower</u>	Power sweep
	<u>SweepTypeSegment</u>	Segment frequency sweep
	<u>ThruAdditionMediaCoaxial</u>	The media of the thru - Specifies the coaxial
	<u>ThruAdditionMediaWaveguide</u>	The media of the thru - Specifies the waveguide
	<u>ThruAdditionUnitMeters</u>	The display units of the thru delay (length) - Selects meters
	<u>ThruAdditionUnitSeconds</u>	The display units of the thru delay (length) - Selects seconds
	<u>TimeDomainGatingBandpass</u>	The gate type of the gating function - Bandpass
	<u>TimeDomainGatingNotch</u>	The gate type of the gating function - Notch

	Name	Description
	TimeDomainGatingShapeMaximum	The gate shape of the gating function - Maximum
	TimeDomainGatingShapeMinimum	The gate shape of the gating function - Minimum
	TimeDomainGatingShapeNormal	The gate shape of the gating function - Normal
	TimeDomainGatingShapeWide	The gate shape of the gating function - Wide
	TimeDomainReflectionOneWay	The reflection distance for the time domain transformation - One Way
	TimeDomainReflectionRoundTrip	The reflection distance for the time domain transformation - Round Trip
	TimeDomainTransformTypeBandpass	The time domain transformation function = Bandpass
	TimeDomainTransformTypeLowpassImpulse	The time domain transformation function = Lowpass Impulse
	TimeDomainTransformTypeLowpassStep	The time domain transformation function = Lowpass Step
	TimeDomainUnitsFeet	The transformation unit for the time domain - Feet
	TimeDomainUnitsMeters	The transformation unit for the time domain - Meters
	TimeDomainUnitsSeconds	The transformation unit for the time domain - Seconds
	TimeDomainWindowShapeArbitrary	The time domain Kaiser–Bessel window shape - Arbitrary

	Name	Description
	<u>TimeDomainWindowShapeMaximum</u>	The time domain Kaiser–Bessel window shape - Maximum
	<u>TimeDomainWindowShapeMinimum</u>	The time domain Kaiser–Bessel window shape - Minimum
	<u>TimeDomainWindowShapeNormal</u>	The time domain Kaiser–Bessel window shape - Normal
	<u>TraceDataMathAdd</u>	The math operation between the data trace and the memory trace - Addition Data + Mem
	<u>TraceDataMathDiv</u>	The math operation between the data trace and the memory trace - Division Data / Mem
	<u>TraceDataMathMult</u>	The math operation between the data trace and the memory trace - Multiplication Data x Mem
	<u>TraceDataMathOff</u>	The math operation between the data trace and the memory trace - No math
	<u>TraceDataMathSubt</u>	The math operation between the data trace and the memory trace - Subtraction Data – Mem
	<u>TraceHoldMax</u>	The type of the trace hold function - Maximum hold
	<u>TraceHoldMin</u>	The type of the trace hold function - Minimum hold
	<u>TraceHoldOff</u>	The type of the trace hold function - Turns off the trace hold function
	<u>TriggerModeContinuous</u>	A sweep actuation occurs every time a trigger signal is detected

	Name	Description
	TriggerModeHold	Sweep actuation is off in the channel, trigger signals do not affect the channel
	TriggerModeSingle	One sweep actuation occurs with trigger signal detection after the mode has been enabled; after the sweep is complete the channel modes changes to hold
	TriggerOutputFunctionAsampling	The trigger output function - After sampling pulse
	TriggerOutputFunctionBsampling	The trigger output function - Before sampling pulse
	TriggerOutputFunctionBsetup	The trigger output function - Before frequency setup pulse
	TriggerOutputFunctionMeasurement	The trigger output function - Measurement sweep signal
	TriggerOutputFunctionReadyForTrigger	The trigger output function - Ready for trigger signal
	TriggerOutputFunctionSweepEnd	The trigger output function - End of sweep pulse
	TriggerOutputPolarityNegative	The polarity of the trigger output - Negative edge
	TriggerOutputPolarityPositive	The polarity of the trigger output - Positive edge
	TriggerRoutePxiTrig0	The connector to use for the external trigger input - Backplane Trigger Line (PXI TRIG0)
	TriggerRoutePxiTrig1	The connector to use for the external trigger input - Backplane Trigger Line (PXI TRIG1)

	Name	Description
	TriggerRoutePxiTrig2	The connector to use for the external trigger input - Backplane Trigger Line (PXI TRIG2)
	TriggerRoutePxiTrig3	The connector to use for the external trigger input - Backplane Trigger Line (PXI TRIG3)
	TriggerRoutePxiTrig4	The connector to use for the external trigger input - Backplane Trigger Line (PXI TRIG4)
	TriggerRoutePxiTrig5	The connector to use for the external trigger input - Backplane Trigger Line (PXI TRIG5)
	TriggerRoutePxiTrig6	The connector to use for the external trigger input - Backplane Trigger Line (PXI TRIG6)
	TriggerRoutePxiTrig7	The connector to use for the external trigger input - Backplane Trigger Line (PXI TRIG7)
	TriggerRouteSmb	The connector to use for the external trigger input - Front panel connector "Ext Trig In"
	TriggerRouteStar	The connector to use for the external trigger input - Backplane Trigger Line (PXI STAR)
	TriggerScopeActiveChannel	The trigger scope - Active channel
	TriggerScopeAllChannel	The trigger scope - All channels
	TriggerSourceBus	The trigger signal is generated by a command communicated from an external computer from a program controlling the Analyzer via automation

	Name	Description
	TriggerSourceExternal	The external trigger input is used as a trigger signal source.
	TriggerSourceInternal	The next trigger signal is generated by the Analyzer on completion of each sweep.
	TriggerSourceManual	The trigger signal is generated by pressing the corresponding softkey.
	UploadTouchstoneFileToActiveTraceMemory	Loads the Touchstone file to the memory trace
	UploadTouchstoneFileToSParameters	Loads the Touchstone file to S-parameters

AdapterRemovalMediaCoaxial

Description

The adapter media - Specifies the coaxial adapter.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AdapterRemovalMediaCoaxial As Integer
```

C#

```
public static int AdapterRemovalMediaCoaxial
```

Visual C++

```
public:  
static int AdapterRemovalMediaCoaxial
```

JavaScript

```
CMT.Instruments.Attributes.adapterRemovalMediaCoaxial
```

Back to [Constants](#)

AdapterRemovalMediaWaveguide

Description

The adapter media - Specifies the waveguide adapter.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AdapterRemovalMediaWaveguide As Integer
```

C#

```
public static int AdapterRemovalMediaWaveguide
```

Visual C++

```
public:  
static int AdapterRemovalMediaWaveguide
```

JavaScript

```
CMT.Instruments.Attributes.adapterRemovalMediaWaveguide
```

Back to [Constants](#)

AdapterRemovalUnitMeters

Description

The display units of the adapter delay - Selects meters.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AdapterRemovalUnitMeters As Integer
```

C#

```
public static int AdapterRemovalUnitMeters
```

Visual C++

```
public:  
static int AdapterRemovalUnitMeters
```

JavaScript

```
CMT.Instruments.Attributes.adapterRemovalUnitMeters
```

Back to [Constants](#)

AdapterRemovalUnitSeconds

Description

The display units of the adapter delay - Selects seconds.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AdapterRemovalUnitSeconds As Integer
```

C#

```
public static int AdapterRemovalUnitSeconds
```

Visual C++

```
public:  
static int AdapterRemovalUnitSeconds
```

JavaScript

```
CMT.Instruments.Attributes.adapterRemovalUnitSeconds
```

Back to [Constants](#)

AnalysisLimitLineMax

Description

The limit line type - Maximum.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisLimitLineMax As Integer
```

C#

```
public static int AnalysisLimitLineMax
```

Visual C++

```
public:  
static int AnalysisLimitLineMax
```

JavaScript

```
CMT.Instruments.Attributes.analysisLimitLineMax
```

Back to [Constants](#)

AnalysisLimitLineMin

Description

The limit line type - Minimum.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisLimitLineMin As Integer
```

C#

```
public static int AnalysisLimitLineMin
```

Visual C++

```
public:  
static int AnalysisLimitLineMin
```

JavaScript

```
CMT.Instruments.Attributes.analysisLimitLineMin
```

Back to [Constants](#)

AnalysisLimitLineOff

Description

The limit line type - Limit Off.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisLimitLineOff As Integer
```

C#

```
public static int AnalysisLimitLineOff
```

Visual C++

```
public:  
static int AnalysisLimitLineOff
```

JavaScript

```
CMT.Instruments.Attributes.analysisLimitLineOff
```

Back to [Constants](#)

AnalysisLimitLineSingle

Description

The limit line type - Single Line.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisLimitLineSingle As Integer
```

C#

```
public static int AnalysisLimitLineSingle
```

Visual C++

```
public:  
static int AnalysisLimitLineSingle
```

JavaScript

```
CMT.Instruments.Attributes.analysisLimitLineSingle
```

Back to [Constants](#)

AnalysisRippleLimitOff

Description

Ripple value display OFF.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisRippleLimitOff As Integer
```

C#

```
public static int AnalysisRippleLimitOff
```

Visual C++

```
public:  
static int AnalysisRippleLimitOff
```

JavaScript

```
CMT.Instruments.Attributes.analysisRippleLimitOff
```

Back to [Constants](#)

AnalysisRippleLimitOn

Description

Ripple value display ON.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisRippleLimitOn As Integer
```

C#

```
public static int AnalysisRippleLimitOn
```

Visual C++

```
public:  
static int AnalysisRippleLimitOn
```

JavaScript

```
CMT.Instruments.Attributes.analysisRippleLimitOn
```

Back to [Constants](#)

AnalysisRippleValueAbsolute

Description

The display type of the ripple value - Absolute value.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisRippleValueAbsolute As Integer
```

C#

```
public static int AnalysisRippleValueAbsolute
```

Visual C++

```
public:  
static int AnalysisRippleValueAbsolute
```

JavaScript

```
CMT.Instruments.Attributes.analysisRippleValueAbsolute
```

Back to [Constants](#)

AnalysisRippleValueMargin

Description

The display type of the ripple value - Margin (difference between the ripple limit and the absolute value).

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisRippleValueMargin As Integer
```

C#

```
public static int AnalysisRippleValueMargin
```

Visual C++

```
public:  
static int AnalysisRippleValueMargin
```

JavaScript

```
CMT.Instruments.Attributes.analysisRippleValueMargin
```

Back to [Constants](#)

AnalysisRippleValueOff

Description

The display type of the ripple value - Ripple value display OFF.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AnalysisRippleValueOff As Integer
```

C#

```
public static int AnalysisRippleValueOff
```

Visual C++

```
public:  
static int AnalysisRippleValueOff
```

JavaScript

```
CMT.Instruments.Attributes.analysisRippleValueOff
```

Back to [Constants](#)

Autocal1port

Description

The type AutoCal Module calibration - Full 1-port calibration.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared Autocal1port As Integer
```

C#

```
public static int Autocal1port
```

Visual C++

```
public:  
static int Autocal1port
```

JavaScript

```
CMT.Instruments.Attributes.autocal1port
```

Back to [Constants](#)

Autocal2port

Description

The type AutoCal Module calibration - Full 2-port calibration.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared Autocal2port As Integer
```

C#

```
public static int Autocal2port
```

Visual C++

```
public:  
static int Autocal2port
```

JavaScript

```
CMT.Instruments.Attributes.autocal2port
```

Back to [Constants](#)

Autocal3port

Description

The type AutoCal Module calibration - Full 3-port calibration.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared Autocal3port As Integer
```

C#

```
public static int Autocal3port
```

Visual C++

```
public:  
static int Autocal3port
```

JavaScript

```
CMT.Instruments.Attributes.autocal3port
```

Back to [Constants](#)

Autocal4port

Description

The type AutoCal Module calibration - Full 4-port calibration.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared Autocal4port As Integer
```

C#

```
public static int Autocal4port
```

Visual C++

```
public:  
static int Autocal4port
```

JavaScript

```
CMT.Instruments.Attributes.autocal4port
```

Back to [Constants](#)

AutocalCharacterizationFactory

Description

The characterization number used when executing AutoCal - Factory characterization.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AutocalCharacterizationFactory As Integer
```

C#

```
public static int AutocalCharacterizationFactory
```

Visual C++

```
public:  
static int AutocalCharacterizationFactory
```

JavaScript

```
CMT.Instruments.Attributes.autocalCharacterizationFactory
```

Back to [Constants](#)

AutocalCharacterizationUser1

Description

The characterization number used when executing AutoCal - User characterization 1.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AutocalCharacterizationUser1 As Integer
```

C#

```
public static int AutocalCharacterizationUser1
```

Visual C++

```
public:  
static int AutocalCharacterizationUser1
```

JavaScript

```
CMT.Instruments.Attributes.autocalCharacterizationUser1
```

Back to [Constants](#)

AutocalCharacterizationUser2

Description

The characterization number used when executing AutoCal - User characterization 2.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AutocalCharacterizationUser2 As Integer
```

C#

```
public static int AutocalCharacterizationUser2
```

Visual C++

```
public:  
static int AutocalCharacterizationUser2
```

JavaScript

```
CMT.Instruments.Attributes.autocalCharacterizationUser2
```

Back to [Constants](#)

AutocalCharacterizationUser3

Description

The characterization number used when executing AutoCal - User characterization 3.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AutocalCharacterizationUser3 As Integer
```

C#

```
public static int AutocalCharacterizationUser3
```

Visual C++

```
public:  
static int AutocalCharacterizationUser3
```

JavaScript

```
CMT.Instruments.Attributes.autocalCharacterizationUser3
```

Back to [Constants](#)

AutocalOnePath2port

Description

The type AutoCal Module calibration - 1 Path 2-port calibration.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AutocalOnePath2port As Integer
```

C#

```
public static int AutocalOnePath2port
```

Visual C++

```
public:  
static int AutocalOnePath2port
```

JavaScript

```
CMT.Instruments.Attributes.autocalOnePath2port
```

Back to [Constants](#)

AutocalOrientationPortA

Description

Port A of AutoCal Module which is connected to a specified port of Network Analyzer.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AutocalOrientationPortA As Integer
```

C#

```
public static int AutocalOrientationPortA
```

Visual C++

```
public:  
static int AutocalOrientationPortA
```

JavaScript

```
CMT.Instruments.Attributes.autocalOrientationPortA
```

Back to [Constants](#)

AutocalOrientationPortB

Description

Port B of AutoCal Module which is connected to a specified port of Network Analyzer.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AutocalOrientationPortB As Integer
```

C#

```
public static int AutocalOrientationPortB
```

Visual C++

```
public:  
static int AutocalOrientationPortB
```

JavaScript

```
CMT.Instruments.Attributes.autocalOrientationPortB
```

Back to [Constants](#)

AutocalOrientationPortC

Description

Port C of AutoCal Module which is connected to a specified port of Network Analyzer.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AutocalOrientationPortC As Integer
```

C#

```
public static int AutocalOrientationPortC
```

Visual C++

```
public:  
static int AutocalOrientationPortC
```

JavaScript

```
CMT.Instruments.Attributes.autocalOrientationPortC
```

Back to [Constants](#)

AutocalOrientationPortD

Description

Port D of AutoCal Module which is connected to a specified port of Network Analyzer.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AutocalOrientationPortD As Integer
```

C#

```
public static int AutocalOrientationPortD
```

Visual C++

```
public:  
static int AutocalOrientationPortD
```

JavaScript

```
CMT.Instruments.Attributes.autocalOrientationPortD
```

Back to [Constants](#)

AutoPortExtensionActiveMarker

Description

Uses the frequency of the active marker for calculation to the Port Extension.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AutoPortExtensionActiveMarker As Integer
```

C#

```
public static int AutoPortExtensionActiveMarker
```

Visual C++

```
public:  
static int AutoPortExtensionActiveMarker
```

JavaScript

```
CMT.Instruments.Attributes.autoPortExtensionActiveMarker
```

Back to [Constants](#)

AutoPortExtensionCurrentSpan

Description

Uses current frequency span for calculation to the Port Extension.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AutoPortExtensionCurrentSpan As Integer
```

C#

```
public static int AutoPortExtensionCurrentSpan
```

Visual C++

```
public:  
static int AutoPortExtensionCurrentSpan
```

JavaScript

```
CMT.Instruments.Attributes.autoPortExtensionCurrentSpan
```

Back to [Constants](#)

AutoPortExtensionUserSpan

Description

Uses arbitrary frequency range set by user for calculation to the Port Extension.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared AutoPortExtensionUserSpan As Integer
```

C#

```
public static int AutoPortExtensionUserSpan
```

Visual C++

```
public:  
static int AutoPortExtensionUserSpan
```

JavaScript

```
CMT.Instruments.Attributes.autoPortExtensionUserSpan
```

Back to [Constants](#)

CalibrationActionIsolation

Description

The type of calibration action - Isolation calibration.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalibrationActionIsolation As Integer
```

C#

```
public static int CalibrationActionIsolation
```

Visual C++

```
public:  
static int CalibrationActionIsolation
```

JavaScript

```
CMT.Instruments.Attributes.calibrationActionIsolation
```

Back to [Constants](#)

CalibrationActionLoad

Description

The type of calibration action - LOAD calibration.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalibrationActionLoad As Integer
```

C#

```
public static int CalibrationActionLoad
```

Visual C++

```
public:  
static int CalibrationActionLoad
```

JavaScript

```
CMT.Instruments.Attributes.calibrationActionLoad
```

Back to [Constants](#)

CalibrationActionOpen

Description

The type of calibration action - OPEN calibration.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalibrationActionOpen As Integer
```

C#

```
public static int CalibrationActionOpen
```

Visual C++

```
public:  
static int CalibrationActionOpen
```

JavaScript

```
CMT.Instruments.Attributes.calibrationActionOpen
```

Back to [Constants](#)

CalibrationActionPowerCal

Description

The type of calibration action - The power calibration.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalibrationActionPowerCal As Integer
```

C#

```
public static int CalibrationActionPowerCal
```

Visual C++

```
public:  
static int CalibrationActionPowerCal
```

JavaScript

```
CMT.Instruments.Attributes.calibrationActionPowerCal
```

Back to [Constants](#)

CalibrationActionReferenceReceiverCal

Description

The type of calibration action - The reference receiver calibration.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalibrationActionReferenceReceiverCal As Integer
```

C#

```
public static int CalibrationActionReferenceReceiverCal
```

Visual C++

```
public:  
static int CalibrationActionReferenceReceiverCal
```

JavaScript

```
CMT.Instruments.Attributes.calibrationActionReferenceReceiverCal
```

Back to [Constants](#)

CalibrationActionShort

Description

The type of calibration action - SHOT calibration.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalibrationActionShort As Integer
```

C#

```
public static int CalibrationActionShort
```

Visual C++

```
public:  
static int CalibrationActionShort
```

JavaScript

```
CMT.Instruments.Attributes.calibrationActionShort
```

Back to [Constants](#)

CalibrationActionTestReceiverCal

Description

The type of calibration action - The test receiver calibration.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalibrationActionTestReceiverCal As Integer
```

C#

```
public static int CalibrationActionTestReceiverCal
```

Visual C++

```
public:  
static int CalibrationActionTestReceiverCal
```

JavaScript

```
CMT.Instruments.Attributes.calibrationActionTestReceiverCal
```

Back to [Constants](#)

CalibrationActionThru

Description

The type of calibration action - THRU calibration.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalibrationActionThru As Integer
```

C#

```
public static int CalibrationActionThru
```

Visual C++

```
public:  
static int CalibrationActionThru
```

JavaScript

```
CMT.Instruments.Attributes.calibrationActionThru
```

Back to [Constants](#)

CalibrationTriggerSourceInternal

Description

The internal trigger source for calibration.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalibrationTriggerSourceInternal As Integer
```

C#

```
public static int CalibrationTriggerSourceInternal
```

Visual C++

```
public:  
static int CalibrationTriggerSourceInternal
```

JavaScript

```
CMT.Instruments.Attributes.calibrationTriggerSourceInternal
```

Back to [Constants](#)

CalibrationTriggerSourceSystem

Description

The system trigger source for calibration.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalibrationTriggerSourceSystem As Integer
```

C#

```
public static int CalibrationTriggerSourceSystem
```

Visual C++

```
public:  
static int CalibrationTriggerSourceSystem
```

JavaScript

```
CMT.Instruments.Attributes.calibrationTriggerSourceSystem
```

Back to [Constants](#)

CalibrationType1path2port

Description

The calibration type for the calculation of the calibration coefficients - One-path 2-port.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalibrationType1path2port As Integer
```

C#

```
public static int CalibrationType1path2port
```

Visual C++

```
public:  
static int CalibrationType1path2port
```

JavaScript

```
CMT.Instruments.Attributes.calibrationType1path2port
```

Back to [Constants](#)

CalibrationType1portSol

Description

The calibration type for the calculation of the calibration coefficients - Full 1-port.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalibrationType1portSol As Integer
```

C#

```
public static int CalibrationType1portSol
```

Visual C++

```
public:  
static int CalibrationType1portSol
```

JavaScript

```
CMT.Instruments.Attributes.calibrationType1portSol
```

Back to [Constants](#)

CalibrationType2portSolt

Description

The calibration type for the calculation of the calibration coefficients - Full 2-port.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalibrationType2portSolt As Integer
```

C#

```
public static int CalibrationType2portSolt
```

Visual C++

```
public:  
static int CalibrationType2portSolt
```

JavaScript

```
CMT.Instruments.Attributes.calibrationType2portSolt
```

Back to [Constants](#)

CalibrationType2portTrl

Description

The calibration type for the calculation of the calibration coefficients - TRL 2-port.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalibrationType2portTrl As Integer
```

C#

```
public static int CalibrationType2portTrl
```

Visual C++

```
public:  
static int CalibrationType2portTrl
```

JavaScript

```
CMT.Instruments.Attributes.calibrationType2portTrl
```

Back to [Constants](#)

CalibrationType3portSolt

Description

The calibration type for the calculation of the calibration coefficients - Full 3-port.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalibrationType3portSolt As Integer
```

C#

```
public static int CalibrationType3portSolt
```

Visual C++

```
public:  
static int CalibrationType3portSolt
```

JavaScript

```
CMT.Instruments.Attributes.calibrationType3portSolt
```

Back to [Constants](#)

CalibrationType3portTrl

Description

The calibration type for the calculation of the calibration coefficients - TRL 3-port.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalibrationType3portTrl As Integer
```

C#

```
public static int CalibrationType3portTrl
```

Visual C++

```
public:  
static int CalibrationType3portTrl
```

JavaScript

```
CMT.Instruments.Attributes.calibrationType3portTrl
```

Back to [Constants](#)

CalibrationType4portSolt

Description

The calibration type for the calculation of the calibration coefficients - Full 4-port.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalibrationType4portSolt As Integer
```

C#

```
public static int CalibrationType4portSolt
```

Visual C++

```
public:  
static int CalibrationType4portSolt
```

JavaScript

```
CMT.Instruments.Attributes.calibrationType4portSolt
```

Back to [Constants](#)

CalibrationType4portTrl

Description

The calibration type for the calculation of the calibration coefficients - TRL 4-port.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalibrationType4portTrl As Integer
```

C#

```
public static int CalibrationType4portTrl
```

Visual C++

```
public:  
static int CalibrationType4portTrl
```

JavaScript

```
CMT.Instruments.Attributes.calibrationType4portTrl
```

Back to [Constants](#)

CalibrationTypeAdapterRemoval

Description

The calibration type for the calculation of the calibration coefficients - Adapter Removal.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalibrationTypeAdapterRemoval As Integer
```

C#

```
public static int CalibrationTypeAdapterRemoval
```

Visual C++

```
public:  
static int CalibrationTypeAdapterRemoval
```

JavaScript

```
CMT.Instruments.Attributes.calibrationTypeAdapterRemoval
```

Back to [Constants](#)

CalibrationTypeNotDefined

Description

The calibration type for the calculation of the calibration coefficients - Not defined.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalibrationTypeNotDefined As Integer
```

C#

```
public static int CalibrationTypeNotDefined
```

Visual C++

```
public:  
static int CalibrationTypeNotDefined
```

JavaScript

```
CMT.Instruments.Attributes.calibrationTypeNotDefined
```

Back to [Constants](#)

CalibrationTypeResponseOpen

Description

The calibration type for the calculation of the calibration coefficients - OPEN response.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalibrationTypeResponseOpen As Integer
```

C#

```
public static int CalibrationTypeResponseOpen
```

Visual C++

```
public:  
static int CalibrationTypeResponseOpen
```

JavaScript

```
CMT.Instruments.Attributes.calibrationTypeResponseOpen
```

Back to [Constants](#)

CalibrationTypeResponseShort

Description

The calibration type for the calculation of the calibration coefficients - SHORT response.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalibrationTypeResponseShort As Integer
```

C#

```
public static int CalibrationTypeResponseShort
```

Visual C++

```
public:  
static int CalibrationTypeResponseShort
```

JavaScript

```
CMT.Instruments.Attributes.calibrationTypeResponseShort
```

Back to [Constants](#)

CalibrationTypeResponseThru

Description

The calibration type for the calculation of the calibration coefficients - THRU response.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalibrationTypeResponseThru As Integer
```

C#

```
public static int CalibrationTypeResponseThru
```

Visual C++

```
public:  
static int CalibrationTypeResponseThru
```

JavaScript

```
CMT.Instruments.Attributes.calibrationTypeResponseThru
```

Back to [Constants](#)

CalkitStandardDataBased

Description

The type of calibration standard - Data Based.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalkitStandardDataBased As Integer
```

C#

```
public static int CalkitStandardDataBased
```

Visual C++

```
public:  
static int CalkitStandardDataBased
```

JavaScript

```
CMT.Instruments.Attributes.calkitStandardDataBased
```

Back to [Constants](#)

CalkitStandardLoad

Description

The type of calibration standard - Load.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalkitStandardLoad As Integer
```

C#

```
public static int CalkitStandardLoad
```

Visual C++

```
public:  
static int CalkitStandardLoad
```

JavaScript

```
CMT.Instruments.Attributes.calkitStandardLoad
```

Back to [Constants](#)

CalkitStandardNone

Description

The type of calibration standard - Not defined.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalkitStandardNone As Integer
```

C#

```
public static int CalkitStandardNone
```

Visual C++

```
public:  
static int CalkitStandardNone
```

JavaScript

```
CMT.Instruments.Attributes.calkitStandardNone
```

Back to [Constants](#)

CalkitStandardOpen

Description

The type of calibration standard - Open.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalkitStandardOpen As Integer
```

C#

```
public static int CalkitStandardOpen
```

Visual C++

```
public:  
static int CalkitStandardOpen
```

JavaScript

```
CMT.Instruments.Attributes.calkitStandardOpen
```

Back to [Constants](#)

CalkitStandardShort

Description

The type of calibration standard - Short.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalkitStandardShort As Integer
```

C#

```
public static int CalkitStandardShort
```

Visual C++

```
public:  
static int CalkitStandardShort
```

JavaScript

```
CMT.Instruments.Attributes.calkitStandardShort
```

Back to [Constants](#)

CalkitStandardSlidingLoad

Description

The type of calibration standard - Sliding Load.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalkitStandardSlidingLoad As Integer
```

C#

```
public static int CalkitStandardSlidingLoad
```

Visual C++

```
public:  
static int CalkitStandardSlidingLoad
```

JavaScript

```
CMT.Instruments.Attributes.calkitStandardSlidingLoad
```

Back to [Constants](#)

CalkitStandardThruDelay

Description

The type of calibration standard - Thru.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalkitStandardThruDelay As Integer
```

C#

```
public static int CalkitStandardThruDelay
```

Visual C++

```
public:  
static int CalkitStandardThruDelay
```

JavaScript

```
CMT.Instruments.Attributes.calkitStandardThruDelay
```

Back to [Constants](#)

CalkitStandardUnknThru

Description

The type of calibration standard - Unknown Thru.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CalkitStandardUnknThru As Integer
```

C#

```
public static int CalkitStandardUnknThru
```

Visual C++

```
public:  
static int CalkitStandardUnknThru
```

JavaScript

```
CMT.Instruments.Attributes.calkitStandardUnknThru
```

Back to [Constants](#)

ChannelStateRegisterA

Description

Saves the Analyzer state of the active channel into memory register A.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ChannelStateRegisterA As Integer
```

C#

```
public static int ChannelStateRegisterA
```

Visual C++

```
public:  
static int ChannelStateRegisterA
```

JavaScript

```
CMT.Instruments.Attributes.channelStateRegisterA
```

Back to [Constants](#)

ChannelStateRegisterB

Description

Saves the Analyzer state of the active channel into memory register B.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ChannelStateRegisterB As Integer
```

C#

```
public static int ChannelStateRegisterB
```

Visual C++

```
public:  
static int ChannelStateRegisterB
```

JavaScript

```
CMT.Instruments.Attributes.channelStateRegisterB
```

Back to [Constants](#)

ChannelStateRegisterC

Description

Saves the Analyzer state of the active channel into memory register C.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ChannelStateRegisterC As Integer
```

C#

```
public static int ChannelStateRegisterC
```

Visual C++

```
public:  
static int ChannelStateRegisterC
```

JavaScript

```
CMT.Instruments.Attributes.channelStateRegisterC
```

Back to [Constants](#)

ChannelStateRegisterD

Description

Saves the Analyzer state of the active channel into memory register D.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ChannelStateRegisterD As Integer
```

C#

```
public static int ChannelStateRegisterD
```

Visual C++

```
public:  
static int ChannelStateRegisterD
```

JavaScript

```
CMT.Instruments.Attributes.channelStateRegisterD
```

Back to [Constants](#)

ConversionFunctionConjugation

Description

S-parameter conversion function type - S-parameter conjugate.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ConversionFunctionConjugation As Integer
```

C#

```
public static int ConversionFunctionConjugation
```

Visual C++

```
public:  
static int ConversionFunctionConjugation
```

JavaScript

```
CMT.Instruments.Attributes.conversionFunctionConjugation
```

Back to [Constants](#)

ConversionFunctionInverse

Description

S-parameter conversion function type - Inverse S-parameter.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ConversionFunctionInverse As Integer
```

C#

```
public static int ConversionFunctionInverse
```

Visual C++

```
public:  
static int ConversionFunctionInverse
```

JavaScript

```
CMT.Instruments.Attributes.conversionFunctionInverse
```

Back to [Constants](#)

ConversionFunctionYReflection

Description

S-parameter conversion function type - Reflection equivalent admittance.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ConversionFunctionYReflection As Integer
```

C#

```
public static int ConversionFunctionYReflection
```

Visual C++

```
public:  
static int ConversionFunctionYReflection
```

JavaScript

```
CMT.Instruments.Attributes.conversionFunctionYReflection
```

Back to [Constants](#)

ConversionFunctionYTransmission

Description

S-parameter conversion function type - Transmission equivalent admittance.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ConversionFunctionYTransmission As Integer
```

C#

```
public static int ConversionFunctionYTransmission
```

Visual C++

```
public:  
static int ConversionFunctionYTransmission
```

JavaScript

```
CMT.Instruments.Attributes.conversionFunctionYTransmission
```

Back to [Constants](#)

ConversionFunctionYTransShunt

Description

S-parameter conversion function type - Shunt equivalent admittance.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ConversionFunctionYTransShunt As Integer
```

C#

```
public static int ConversionFunctionYTransShunt
```

Visual C++

```
public:  
static int ConversionFunctionYTransShunt
```

JavaScript

```
CMT.Instruments.Attributes.conversionFunctionYTransShunt
```

Back to [Constants](#)

ConversionFunctionZReflection

Description

S-parameter conversion function type - Reflection equivalent impedance.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ConversionFunctionZReflection As Integer
```

C#

```
public static int ConversionFunctionZReflection
```

Visual C++

```
public:  
static int ConversionFunctionZReflection
```

JavaScript

```
CMT.Instruments.Attributes.conversionFunctionZReflection
```

Back to [Constants](#)

ConversionFunctionZTransmission

Description

S-parameter conversion function type - Transmission equivalent impedance.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ConversionFunctionZTransmission As Integer
```

C#

```
public static int ConversionFunctionZTransmission
```

Visual C++

```
public:  
static int ConversionFunctionZTransmission
```

JavaScript

```
CMT.Instruments.Attributes.conversionFunctionZTransmission
```

Back to [Constants](#)

ConversionFunctionZTransShunt

Description

S-parameter conversion function type - Shunt equivalent impedance.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ConversionFunctionZTransShunt As Integer
```

C#

```
public static int ConversionFunctionZTransShunt
```

Visual C++

```
public:  
static int ConversionFunctionZTransShunt
```

JavaScript

```
CMT.Instruments.Attributes.conversionFunctionZTransShunt
```

Back to [Constants](#)

CorrectionStateCorrection

Description

Correction status of the error correction.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CorrectionStateCorrection As Integer
```

C#

```
public static int CorrectionStateCorrection
```

Visual C++

```
public:  
static int CorrectionStateCorrection
```

JavaScript

```
CMT.Instruments.Attributes.correctionStateCorrection
```

Back to [Constants](#)

CorrectionStateExtrapolation

Description

Extrapolation status of the error correction.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CorrectionStateExtrapolation As Integer
```

C#

```
public static int CorrectionStateExtrapolation
```

Visual C++

```
public:  
static int CorrectionStateExtrapolation
```

JavaScript

```
CMT.Instruments.Attributes.correctionStateExtrapolation
```

Back to [Constants](#)

CorrectionStateInterpolation

Description

Interpolation status of the error correction.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CorrectionStateInterpolation As Integer
```

C#

```
public static int CorrectionStateInterpolation
```

Visual C++

```
public:  
static int CorrectionStateInterpolation
```

JavaScript

```
CMT.Instruments.Attributes.correctionStateInterpolation
```

Back to [Constants](#)

CorrectionStateNone

Description

None status of the error correction.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared CorrectionStateNone As Integer
```

C#

```
public static int CorrectionStateNone
```

Visual C++

```
public:  
static int CorrectionStateNone
```

JavaScript

```
CMT.Instruments.Attributes.correctionStateNone
```

Back to [Constants](#)

DisplayTraceData

Description

Only data trace is displayed.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DisplayTraceData As Integer
```

C#

```
public static int DisplayTraceData
```

Visual C++

```
public:  
static int DisplayTraceData
```

JavaScript

```
CMT.Instruments.Attributes.displayTraceData
```

Back to [Constants](#)

DisplayTraceDataAndMemory

Description

Data trace and memory trace are displayed.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DisplayTraceDataAndMemory As Integer
```

C#

```
public static int DisplayTraceDataAndMemory
```

Visual C++

```
public:  
static int DisplayTraceDataAndMemory
```

JavaScript

```
CMT.Instruments.Attributes.displayTraceDataAndMemory
```

Back to [Constants](#)

DisplayTraceMemory

Description

Only memory trace is displayed.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DisplayTraceMemory As Integer
```

C#

```
public static int DisplayTraceMemory
```

Visual C++

```
public:  
static int DisplayTraceMemory
```

JavaScript

```
CMT.Instruments.Attributes.displayTraceMemory
```

Back to [Constants](#)

DisplayTraceOff

Description

Both traces are not displayed.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared DisplayTraceOff As Integer
```

C#

```
public static int DisplayTraceOff
```

Visual C++

```
public:  
static int DisplayTraceOff
```

JavaScript

```
CMT.Instruments.Attributes.displayTraceOff
```

Back to [Constants](#)

ExtTriggerEventOnPoint

Description

The external trigger response is the single point.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ExtTriggerEventOnPoint As Integer
```

C#

```
public static int ExtTriggerEventOnPoint
```

Visual C++

```
public:  
static int ExtTriggerEventOnPoint
```

JavaScript

```
CMT.Instruments.Attributes.extTriggerEventOnPoint
```

Back to [Constants](#)

ExtTriggerEventOnSweep

Description

The external trigger response is the entire sweep.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ExtTriggerEventOnSweep As Integer
```

C#

```
public static int ExtTriggerEventOnSweep
```

Visual C++

```
public:  
static int ExtTriggerEventOnSweep
```

JavaScript

```
CMT.Instruments.Attributes.extTriggerEventOnSweep
```

Back to [Constants](#)

ExtTriggerPolarityNegative

Description

The polarity of the external trigger - Negative edge.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ExtTriggerPolarityNegative As Integer
```

C#

```
public static int ExtTriggerPolarityNegative
```

Visual C++

```
public:  
static int ExtTriggerPolarityNegative
```

JavaScript

```
CMT.Instruments.Attributes.extTriggerPolarityNegative
```

Back to [Constants](#)

ExtTriggerPolarityPositive

Description

The polarity of the external trigger - Positive edge.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ExtTriggerPolarityPositive As Integer
```

C#

```
public static int ExtTriggerPolarityPositive
```

Visual C++

```
public:  
static int ExtTriggerPolarityPositive
```

JavaScript

```
CMT.Instruments.Attributes.extTriggerPolarityPositive
```

Back to [Constants](#)

ExtTriggerPositionBsampling

Description

The position of the external trigger - Before sampling.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ExtTriggerPositionBsampling As Integer
```

C#

```
public static int ExtTriggerPositionBsampling
```

Visual C++

```
public:  
static int ExtTriggerPositionBsampling
```

JavaScript

```
CMT.Instruments.Attributes.extTriggerPositionBsampling
```

Back to [Constants](#)

ExtTriggerPositionBsetup

Description

The position of the external trigger - Before frequency setup.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ExtTriggerPositionBsetup As Integer
```

C#

```
public static int ExtTriggerPositionBsetup
```

Visual C++

```
public:  
static int ExtTriggerPositionBsetup
```

JavaScript

```
CMT.Instruments.Attributes.extTriggerPositionBsetup
```

Back to [Constants](#)

FrequencyOffsetTypePortPort

Description

The frequency offset type - Port1/Port2 offset.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FrequencyOffsetTypePortPort As Integer
```

C#

```
public static int FrequencyOffsetTypePortPort
```

Visual C++

```
public:  
static int FrequencyOffsetTypePortPort
```

JavaScript

```
CMT.Instruments.Attributes.frequencyOffsetTypePortPort
```

Back to [Constants](#)

FrequencyOffsetTypeSourceReceiver

Description

The frequency offset type - Source/Receivers offset.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FrequencyOffsetTypeSourceReceiver As Integer
```

C#

```
public static int FrequencyOffsetTypeSourceReceiver
```

Visual C++

```
public:  
static int FrequencyOffsetTypeSourceReceiver
```

JavaScript

```
CMT.Instruments.Attributes.frequencyOffsetTypeSourceReceiver
```

Back to [Constants](#)

FunctionAllPeaks

Description

The type of analysis - Search for all the peaks.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FunctionAllPeaks As Integer
```

C#

```
public static int FunctionAllPeaks
```

Visual C++

```
public:  
static int FunctionAllPeaks
```

JavaScript

```
CMT.Instruments.Attributes.functionAllPeaks
```

Back to [Constants](#)

FunctionAllTarget

Description

The type of analysis - Search for all targets.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FunctionAllTarget As Integer
```

C#

```
public static int FunctionAllTarget
```

Visual C++

```
public:  
static int FunctionAllTarget
```

JavaScript

```
CMT.Instruments.Attributes.functionAllTarget
```

Back to [Constants](#)

FunctionMaximum

Description

The type of analysis - Maximum value.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FunctionMaximum As Integer
```

C#

```
public static int FunctionMaximum
```

Visual C++

```
public:  
static int FunctionMaximum
```

JavaScript

```
CMT.Instruments.Attributes.functionMaximum
```

Back to [Constants](#)

FunctionMean

Description

The type of analysis - Mean value.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FunctionMean As Integer
```

C#

```
public static int FunctionMean
```

Visual C++

```
public:  
static int FunctionMean
```

JavaScript

```
CMT.Instruments.Attributes.functionMean
```

Back to [Constants](#)

FunctionMinimum

Description

The type of analysis - Minimum value.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FunctionMinimum As Integer
```

C#

```
public static int FunctionMinimum
```

Visual C++

```
public:  
static int FunctionMinimum
```

JavaScript

```
CMT.Instruments.Attributes.functionMinimum
```

Back to [Constants](#)

FunctionPeak

Description

The type of analysis - Search for peak.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FunctionPeak As Integer
```

C#

```
public static int FunctionPeak
```

Visual C++

```
public:  
static int FunctionPeak
```

JavaScript

```
CMT.Instruments.Attributes.functionPeak
```

Back to [Constants](#)

FunctionPeakPolarityBoth

Description

The polarity when performing the peak search - Both: positive peaks and negative peaks.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FunctionPeakPolarityBoth As Integer
```

C#

```
public static int FunctionPeakPolarityBoth
```

Visual C++

```
public:  
static int FunctionPeakPolarityBoth
```

JavaScript

```
CMT.Instruments.Attributes.functionPeakPolarityBoth
```

Back to [Constants](#)

FunctionPeakPolarityNegative

Description

The polarity when performing the peak search - Negative peaks.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FunctionPeakPolarityNegative As Integer
```

C#

```
public static int FunctionPeakPolarityNegative
```

Visual C++

```
public:  
static int FunctionPeakPolarityNegative
```

JavaScript

```
CMT.Instruments.Attributes.functionPeakPolarityNegative
```

Back to [Constants](#)

FunctionPeakPolarityPositive

Description

The polarity when performing the peak search - Positive peaks.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FunctionPeakPolarityPositive As Integer
```

C#

```
public static int FunctionPeakPolarityPositive
```

Visual C++

```
public:  
static int FunctionPeakPolarityPositive
```

JavaScript

```
CMT.Instruments.Attributes.functionPeakPolarityPositive
```

Back to [Constants](#)

FunctionPeakToPeak

Description

The type of analysis - Peak-to-peak (difference between the maximum value and the minimum value).

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FunctionPeakToPeak As Integer
```

C#

```
public static int FunctionPeakToPeak
```

Visual C++

```
public:  
static int FunctionPeakToPeak
```

JavaScript

```
CMT.Instruments.Attributes.functionPeakToPeak
```

Back to [Constants](#)

FunctionStandardDeviation

Description

The type of analysis - Standard deviation.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FunctionStandardDeviation As Integer
```

C#

```
public static int FunctionStandardDeviation
```

Visual C++

```
public:  
static int FunctionStandardDeviation
```

JavaScript

```
CMT.Instruments.Attributes.functionStandardDeviation
```

Back to [Constants](#)

FunctionTransitionTypeBoth

Description

The transition type when performing the search for the trace - Positive and Negative.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FunctionTransitionTypeBoth As Integer
```

C#

```
public static int FunctionTransitionTypeBoth
```

Visual C++

```
public:  
static int FunctionTransitionTypeBoth
```

JavaScript

```
CMT.Instruments.Attributes.functionTransitionTypeBoth
```

Back to [Constants](#)

FunctionTransitionTypeNegative

Description

The transition type when performing the search for the trace - Negative.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FunctionTransitionTypeNegative As Integer
```

C#

```
public static int FunctionTransitionTypeNegative
```

Visual C++

```
public:  
static int FunctionTransitionTypeNegative
```

JavaScript

```
CMT.Instruments.Attributes.functionTransitionTypeNegative
```

Back to [Constants](#)

FunctionTransitionTypePositive

Description

The transition type when performing the search for the trace - Positive.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared FunctionTransitionTypePositive As Integer
```

C#

```
public static int FunctionTransitionTypePositive
```

Visual C++

```
public:  
static int FunctionTransitionTypePositive
```

JavaScript

```
CMT.Instruments.Attributes.functionTransitionTypePositive
```

Back to [Constants](#)

MarkerMathBandwidthSearchBandpass

Description

The type of the bandwidth search function - Bandpass.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerMathBandwidthSearchBandpass As Integer
```

C#

```
public static int MarkerMathBandwidthSearchBandpass
```

Visual C++

```
public:  
static int MarkerMathBandwidthSearchBandpass
```

JavaScript

```
CMT.Instruments.Attributes.markerMathBandwidthSearchBandpass
```

Back to [Constants](#)

MarkerMathBandwidthSearchNotch

Description

The type of the bandwidth search function - Notch.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerMathBandwidthSearchNotch As Integer
```

C#

```
public static int MarkerMathBandwidthSearchNotch
```

Visual C++

```
public:  
static int MarkerMathBandwidthSearchNotch
```

JavaScript

```
CMT.Instruments.Attributes.markerMathBandwidthSearchNotch
```

Back to [Constants](#)

MarkerMathBandwidthSearchRefMarker

Description

Bandwidth search relative to the reference marker.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerMathBandwidthSearchRefMarker As Integer
```

C#

```
public static int MarkerMathBandwidthSearchRefMarker
```

Visual C++

```
public:  
static int MarkerMathBandwidthSearchRefMarker
```

JavaScript

```
CMT.Instruments.Attributes.markerMathBandwidthSearchRefMarker
```

Back to [Constants](#)

MarkerMathBandwidthSearchRefMinimum

Description

Bandwidth search relative to the absolute minimum of the trace.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerMathBandwidthSearchRefMinimum As Integer
```

C#

```
public static int MarkerMathBandwidthSearchRefMinimum
```

Visual C++

```
public:  
static int MarkerMathBandwidthSearchRefMinimum
```

JavaScript

```
CMT.Instruments.Attributes.markerMathBandwidthSearchRefMinimum
```

Back to [Constants](#)

MarkerMathBandwidthSearchRefMaximum

Description

Bandwidth search relative to the absolute maximum of the trace.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerMathBandwidthSearchRefMaximum As Integer
```

C#

```
public static int MarkerMathBandwidthSearchRefMaximum
```

Visual C++

```
public:  
static int MarkerMathBandwidthSearchRefMaximum
```

JavaScript

```
CMT.Instruments.Constants.markerMathBandwidthSearchRefMaximum
```

Back to [Constants](#)

MarkerPropertiesAlignHorizontal

Description

The alignment mode of the marker display position - Horizontal.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerPropertiesAlignHorizontal As Integer
```

C#

```
public static int MarkerPropertiesAlignHorizontal
```

Visual C++

```
public:  
static int MarkerPropertiesAlignHorizontal
```

JavaScript

```
CMT.Instruments.Attributes.markerPropertiesAlignHorizontal
```

Back to [Constants](#)

MarkerPropertiesAlignOff

Description

The alignment mode of the marker display position - No alignment.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerPropertiesAlignOff As Integer
```

C#

```
public static int MarkerPropertiesAlignOff
```

Visual C++

```
public:  
static int MarkerPropertiesAlignOff
```

JavaScript

```
CMT.Instruments.Attributes.markerPropertiesAlignOff
```

Back to [Constants](#)

MarkerPropertiesAlignVertical

Description

The alignment mode of the marker display position - Vertical.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerPropertiesAlignVertical As Integer
```

C#

```
public static int MarkerPropertiesAlignVertical
```

Visual C++

```
public:  
static int MarkerPropertiesAlignVertical
```

JavaScript

```
CMT.Instruments.Attributes.markerPropertiesAlignVertical
```

Back to [Constants](#)

MarkerSearchPeakPolarityBoth

Description

The peak polarity, when the marker search for peak - Both positive polarity and negative polarity.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerSearchPeakPolarityBoth As Integer
```

C#

```
public static int MarkerSearchPeakPolarityBoth
```

Visual C++

```
public:  
static int MarkerSearchPeakPolarityBoth
```

JavaScript

```
CMT.Instruments.Attributes.markerSearchPeakPolarityBoth
```

Back to [Constants](#)

MarkerSearchPeakPolarityNegative

Description

The peak polarity, when the marker search for peak - Negative polarity.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerSearchPeakPolarityNegative As Integer
```

C#

```
public static int MarkerSearchPeakPolarityNegative
```

Visual C++

```
public:  
static int MarkerSearchPeakPolarityNegative
```

JavaScript

```
CMT.Instruments.Attributes.markerSearchPeakPolarityNegative
```

Back to [Constants](#)

MarkerSearchPeakPolarityPositive

Description

The peak polarity, when the marker search for peak - Positive polarity.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerSearchPeakPolarityPositive As Integer
```

C#

```
public static int MarkerSearchPeakPolarityPositive
```

Visual C++

```
public:  
static int MarkerSearchPeakPolarityPositive
```

JavaScript

```
CMT.Instruments.Attributes.markerSearchPeakPolarityPositive
```

Back to [Constants](#)

MarkerSearchTargetTransitionBoth

Description

The type of the target transition, when the marker search for transition - Both positive target transition and negative target transition.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerSearchTargetTransitionBoth As Integer
```

C#

```
public static int MarkerSearchTargetTransitionBoth
```

Visual C++

```
public:  
static int MarkerSearchTargetTransitionBoth
```

JavaScript

```
CMT.Instruments.Attributes.markerSearchTargetTransitionBoth
```

Back to [Constants](#)

MarkerSearchTargetTransitionNegative

Description

The type of the target transition, when the marker search for transition - Negative target transition.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerSearchTargetTransitionNegative As Integer
```

C#

```
public static int MarkerSearchTargetTransitionNegative
```

Visual C++

```
public:  
static int MarkerSearchTargetTransitionNegative
```

JavaScript

```
CMT.Instruments.Attributes.markerSearchTargetTransitionNegative
```

Back to [Constants](#)

MarkerSearchTargetTransitionPositive

Description

The type of the target transition, when the marker search for transition - Positive target transition.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerSearchTargetTransitionPositive As Integer
```

C#

```
public static int MarkerSearchTargetTransitionPositive
```

Visual C++

```
public:  
static int MarkerSearchTargetTransitionPositive
```

JavaScript

```
CMT.Instruments.Attributes.markerSearchTargetTransitionPositive
```

Back to [Constants](#)

MarkerSearchTypeMaximum

Description

The type of the marker search - Maximum.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerSearchTypeMaximum As Integer
```

C#

```
public static int MarkerSearchTypeMaximum
```

Visual C++

```
public:  
static int MarkerSearchTypeMaximum
```

JavaScript

```
CMT.Instruments.Attributes.markerSearchTypeMaximum
```

Back to [Constants](#)

MarkerSearchTypeMinimum

Description

The type of the marker search - Minimum.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerSearchTypeMinimum As Integer
```

C#

```
public static int MarkerSearchTypeMinimum
```

Visual C++

```
public:  
static int MarkerSearchTypeMinimum
```

JavaScript

```
CMT.Instruments.Attributes.markerSearchTypeMinimum
```

Back to [Constants](#)

MarkerSearchTypePeak

Description

The type of the marker search - Search Peak.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerSearchTypePeak As Integer
```

C#

```
public static int MarkerSearchTypePeak
```

Visual C++

```
public:  
static int MarkerSearchTypePeak
```

JavaScript

```
CMT.Instruments.Attributes.markerSearchTypePeak
```

Back to [Constants](#)

MarkerSearchTypePeakLeft

Description

The type of the marker search - Search Peak Left.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerSearchTypePeakLeft As Integer
```

C#

```
public static int MarkerSearchTypePeakLeft
```

Visual C++

```
public:  
static int MarkerSearchTypePeakLeft
```

JavaScript

```
CMT.Instruments.Attributes.markerSearchTypePeakLeft
```

Back to [Constants](#)

MarkerSearchTypePeakRight

Description

The type of the marker search - Search Peak Right.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerSearchTypePeakRight As Integer
```

C#

```
public static int MarkerSearchTypePeakRight
```

Visual C++

```
public:  
static int MarkerSearchTypePeakRight
```

JavaScript

```
CMT.Instruments.Attributes.markerSearchTypePeakRight
```

Back to [Constants](#)

MarkerSearchTypeTarget

Description

The type of the marker search - Search Target.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerSearchTypeTarget As Integer
```

C#

```
public static int MarkerSearchTypeTarget
```

Visual C++

```
public:  
static int MarkerSearchTypeTarget
```

JavaScript

```
CMT.Instruments.Attributes.markerSearchTypeTarget
```

Back to [Constants](#)

MarkerSearchTypeTargetLeft

Description

The type of the marker search - Search Target Left.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerSearchTypeTargetLeft As Integer
```

C#

```
public static int MarkerSearchTypeTargetLeft
```

Visual C++

```
public:  
static int MarkerSearchTypeTargetLeft
```

JavaScript

```
CMT.Instruments.Attributes.markerSearchTypeTargetLeft
```

Back to [Constants](#)

MarkerSearchTypeTargetRight

Description

The type of the marker search - Search Target Right.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MarkerSearchTypeTargetRight As Integer
```

C#

```
public static int MarkerSearchTypeTargetRight
```

Visual C++

```
public:  
static int MarkerSearchTypeTargetRight
```

JavaScript

```
CMT.Instruments.Attributes.markerSearchTypeTargetRight
```

Back to [Constants](#)

MeasurementFormatExpandPhase

Description

Measurement Format - Expanded phase.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementFormatExpandPhase As Integer
```

C#

```
public static int MeasurementFormatExpandPhase
```

Visual C++

```
public:  
static int MeasurementFormatExpandPhase
```

JavaScript

```
CMT.Instruments.Attributes.measurementFormatExpandPhase
```

Back to [Constants](#)

MeasurementFormatGroupDelay

Description

Measurement Format - Group delay time.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementFormatGroupDelay As Integer
```

C#

```
public static int MeasurementFormatGroupDelay
```

Visual C++

```
public:  
static int MeasurementFormatGroupDelay
```

JavaScript

```
CMT.Instruments.Attributes.measurementFormatGroupDelay
```

Back to [Constants](#)

MeasurementFormatImag

Description

Measurement Format - Imaginary part.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementFormatImag As Integer
```

C#

```
public static int MeasurementFormatImag
```

Visual C++

```
public:  
static int MeasurementFormatImag
```

JavaScript

```
CMT.Instruments.Attributes.measurementFormatImag
```

Back to [Constants](#)

MeasurementFormatLinMag

Description

Measurement Format - Linear magnitude.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementFormatLinMag As Integer
```

C#

```
public static int MeasurementFormatLinMag
```

Visual C++

```
public:  
static int MeasurementFormatLinMag
```

JavaScript

```
CMT.Instruments.Attributes.measurementFormatLinMag
```

Back to [Constants](#)

MeasurementFormatLogMag

Description

Measurement Format - Logarithmic magnitude.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementFormatLogMag As Integer
```

C#

```
public static int MeasurementFormatLogMag
```

Visual C++

```
public:  
static int MeasurementFormatLogMag
```

JavaScript

```
CMT.Instruments.Attributes.measurementFormatLogMag
```

Back to [Constants](#)

MeasurementFormatPhase

Description

Measurement Format - Phase.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementFormatPhase As Integer
```

C#

```
public static int MeasurementFormatPhase
```

Visual C++

```
public:  
static int MeasurementFormatPhase
```

JavaScript

```
CMT.Instruments.Attributes.measurementFormatPhase
```

Back to [Constants](#)

MeasurementFormatPolarLin

Description

Measurement Format - Polar format (Lin).

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementFormatPolarLin As Integer
```

C#

```
public static int MeasurementFormatPolarLin
```

Visual C++

```
public:  
static int MeasurementFormatPolarLin
```

JavaScript

```
CMT.Instruments.Attributes.measurementFormatPolarLin
```

Back to [Constants](#)

MeasurementFormatPolarLog

Description

Measurement Format - Polar format (Log).

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementFormatPolarLog As Integer
```

C#

```
public static int MeasurementFormatPolarLog
```

Visual C++

```
public:  
static int MeasurementFormatPolarLog
```

JavaScript

```
CMT.Instruments.Attributes.measurementFormatPolarLog
```

Back to [Constants](#)

MeasurementFormatPolarReallImage

Description

Measurement Format - Polar format (Real/Imag).

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementFormatPolarReallImage As Integer
```

C#

```
public static int MeasurementFormatPolarReallImage
```

Visual C++

```
public:  
static int MeasurementFormatPolarReallImage
```

JavaScript

```
CMT.Instruments.Attributes.measurementFormatPolarReallImage
```

Back to [Constants](#)

MeasurementFormatReal

Description

Measurement Format - Real part.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementFormatReal As Integer
```

C#

```
public static int MeasurementFormatReal
```

Visual C++

```
public:  
static int MeasurementFormatReal
```

JavaScript

```
CMT.Instruments.Attributes.measurementFormatReal
```

Back to [Constants](#)

MeasurementFormatSmithAdmittance

Description

Measurement Format - Smith chart format ($G + jB$).

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementFormatSmithAdmittance As Integer
```

C#

```
public static int MeasurementFormatSmithAdmittance
```

Visual C++

```
public:  
static int MeasurementFormatSmithAdmittance
```

JavaScript

```
CMT.Instruments.Attributes.measurementFormatSmithAdmittance
```

Back to [Constants](#)

MeasurementFormatSmithComplex

Description

Measurement Format - Smith chart format ($R + jX$)

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementFormatSmithComplex As Integer
```

C#

```
public static int MeasurementFormatSmithComplex
```

Visual C++

```
public:  
static int MeasurementFormatSmithComplex
```

JavaScript

```
CMT.Instruments.Attributes.measurementFormatSmithComplex
```

Back to [Constants](#)

MeasurementFormatSmithLin

Description

Measurement Format - Smith chart format (Lin).

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementFormatSmithLin As Integer
```

C#

```
public static int MeasurementFormatSmithLin
```

Visual C++

```
public:  
static int MeasurementFormatSmithLin
```

JavaScript

```
CMT.Instruments.Attributes.measurementFormatSmithLin
```

Back to [Constants](#)

MeasurementFormatSmithLog

Description

Measurement Format - Smith chart format (Log).

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementFormatSmithLog As Integer
```

C#

```
public static int MeasurementFormatSmithLog
```

Visual C++

```
public:  
static int MeasurementFormatSmithLog
```

JavaScript

```
CMT.Instruments.Constants.measurementFormatSmithLog
```

Back to [Constants](#)

MeasurementFormatSmithReallImage

Description

Measurement Format - Smith chart format (Real/Imag).

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementFormatSmithReallImage As Integer
```

C#

```
public static int MeasurementFormatSmithReallImage
```

Visual C++

```
public:  
static int MeasurementFormatSmithReallImage
```

JavaScript

```
CMT.Instruments.Attributes.measurementFormatSmithReallImage
```

Back to [Constants](#)

MeasurementTypeAbsoluteR

Description

The measurement parameter of the trace - Reference receiver: R1, R2, R3, R4.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementTypeAbsoluteR As Integer
```

C#

```
public static int MeasurementTypeAbsoluteR
```

Visual C++

```
public:  
static int MeasurementTypeAbsoluteR
```

JavaScript

```
CMT.Instruments.Attributes.measurementTypeAbsoluteR
```

Back to [Constants](#)

MeasurementFormatSwr

Description

Measurement Format - Voltage standing wave ratio.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementFormatSwr As Integer
```

C#

```
public static int MeasurementFormatSwr
```

Visual C++

```
public:  
static int MeasurementFormatSwr
```

JavaScript

```
CMT.Instruments.Attributes.measurementFormatSwr
```

Back to [Constants](#)

MeasurementTypeAbsoluteT

Description

The measurement parameter of the trace - Test receiver: A, B, C, D or T1, T2, T3, T4.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared MeasurementTypeAbsoluteT As Integer
```

C#

```
public static int MeasurementTypeAbsoluteT
```

Visual C++

```
public:  
static int MeasurementTypeAbsoluteT
```

JavaScript

```
CMT.Instruments.Attributes.measurementTypeAbsoluteT
```

Back to [Constants](#)

PowerSensorNrpxt

Description

The power sensor type to be used in a source power calibration - Rohde_Schwarz NRPxT series Sensors.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared PowerSensorNrpxt As Integer
```

C#

```
public static int PowerSensorNrpxt
```

Visual C++

```
public:  
static int PowerSensorNrpxt
```

JavaScript

```
CMT.Instruments.Attributes.powerSensorNrpxt
```

Back to [Constants](#)

PowerSensorNrpz

Description

The power sensor type to be used in a source power calibration - Rohde_Schwarz NRP-Z series Sensors.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared PowerSensorNrpz As Integer
```

C#

```
public static int PowerSensorNrpz
```

Visual C++

```
public:  
static int PowerSensorNrpz
```

JavaScript

```
CMT.Instruments.Attributes.powerSensorNrpz
```

Back to [Constants](#)

PowerSensorNrvs

Description

The power sensor type to be used in a source power calibration - Rohde_Schwarz NRVS power meter.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared PowerSensorNrvs As Integer
```

C#

```
public static int PowerSensorNrvs
```

Visual C++

```
public:  
static int PowerSensorNrvs
```

JavaScript

```
CMT.Instruments.Attributes.powerSensorNrvs
```

Back to [Constants](#)

PowerSensorU200x

Description

The power sensor type to be used in a source power calibration - Keysight U200x series Sensors.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared PowerSensorU200x As Integer
```

C#

```
public static int PowerSensorU200x
```

Visual C++

```
public:  
static int PowerSensorU200x
```

JavaScript

```
CMT.Instruments.Attributes.powerSensorU200x
```

Back to [Constants](#)

PowerSensorU848x

Description

The power sensor type to be used in a source power calibration - Keysight U848x series Sensors.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared PowerSensorU848x As Integer
```

C#

```
public static int PowerSensorU848x
```

Visual C++

```
public:  
static int PowerSensorU848x
```

JavaScript

```
CMT.Instruments.Attributes.powerSensorU848x
```

Back to [Constants](#)

PrintBlackAndWhite

Description

The color chart for the image printout - Black and white printout.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared PrintBlackAndWhite As Integer
```

C#

```
public static int PrintBlackAndWhite
```

Visual C++

```
public:  
static int PrintBlackAndWhite
```

JavaScript

```
CMT.Instruments.Attributes.printBlackAndWhite
```

Back to [Constants](#)

PrintColor

Description

The color chart for the image printout - Color printout.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared PrintColor As Integer
```

C#

```
public static int PrintColor
```

Visual C++

```
public:  
static int PrintColor
```

JavaScript

```
CMT.Instruments.Attributes.printColor
```

Back to [Constants](#)

PrintGrayScale

Description

The color chart for the image printout - Grayscale printout.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared PrintGrayScale As Integer
```

C#

```
public static int PrintGrayScale
```

Visual C++

```
public:  
static int PrintGrayScale
```

JavaScript

```
CMT.Instruments.Attributes.printGrayScale
```

Back to [Constants](#)

ReferenceFrequencySourceExternal

Description

External source of the reference frequency of 10 MHz.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ReferenceFrequencySourceExternal As Integer
```

C#

```
public static int ReferenceFrequencySourceExternal
```

Visual C++

```
public:  
static int ReferenceFrequencySourceExternal
```

JavaScript

```
CMT.Instruments.Attributes.referenceFrequencySourceExternal
```

Back to [Constants](#)

ReferenceFrequencySourceInternal

Description

Internal source of the reference frequency of 10 MHz.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ReferenceFrequencySourceInternal As Integer
```

C#

```
public static int ReferenceFrequencySourceInternal
```

Visual C++

```
public:  
static int ReferenceFrequencySourceInternal
```

JavaScript

```
CMT.Instruments.Attributes.referenceFrequencySourceInternal
```

Back to [Constants](#)

ReferenceRouteFront

Description

The route of the external 10 MHz reference frequency - Front panel.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ReferenceRouteFront As Integer
```

C#

```
public static int ReferenceRouteFront
```

Visual C++

```
public:  
static int ReferenceRouteFront
```

JavaScript

```
CMT.Instruments.Attributes.referenceRouteFront
```

Back to [Constants](#)

ReferenceRouteRear

Description

The route of the external 10 MHz reference frequency - Rear panel.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ReferenceRouteRear As Integer
```

C#

```
public static int ReferenceRouteRear
```

Visual C++

```
public:  
static int ReferenceRouteRear
```

JavaScript

```
CMT.Instruments.Attributes.referenceRouteRear
```

Back to [Constants](#)

SaveTouchstoneFileColumnSeparatorSpace

Description

The Touchstone file separator symbol - Space symbol (0x20).

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared SaveTouchstoneFileColumnSeparatorSpace As Integer
```

C#

```
public static int SaveTouchstoneFileColumnSeparatorSpace
```

Visual C++

```
public:  
static int SaveTouchstoneFileColumnSeparatorSpace
```

JavaScript

```
CMT.Instruments.Attributes.saveTouchstoneFileColumnSeparatorSpace
```

Back to [Constants](#)

SaveTouchstoneFileColumnSeparatorTab

Description

The Touchstone file separator symbol - Tab symbol (0x09).

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared SaveTouchstoneFileColumnSeparatorTab As Integer
```

C#

```
public static int SaveTouchstoneFileColumnSeparatorTab
```

Visual C++

```
public:  
static int SaveTouchstoneFileColumnSeparatorTab
```

JavaScript

```
CMT.Instruments.Attributes.saveTouchstoneFileColumnSeparatorTab
```

Back to [Constants](#)

SaveTouchstoneFileFormatDbAngle

Description

The data format for the S-parameter saving - Db / Angle.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared SaveTouchstoneFileFormatDbAngle As Integer
```

C#

```
public static int SaveTouchstoneFileFormatDbAngle
```

Visual C++

```
public:  
static int SaveTouchstoneFileFormatDbAngle
```

JavaScript

```
CMT.Instruments.Attributes.saveTouchstoneFileFormatDbAngle
```

Back to [Constants](#)

SaveTouchstoneFileFormatMagnitudeAngle

Description

The data format for the S-parameter saving - Magnitude / Angle.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared SaveTouchstoneFileFormatMagnitudeAngle As Integer
```

C#

```
public static int SaveTouchstoneFileFormatMagnitudeAngle
```

Visual C++

```
public:  
static int SaveTouchstoneFileFormatMagnitudeAngle
```

JavaScript

```
CMT.Instruments.Attributes.saveTouchstoneFileFormatMagnitudeAngle
```

Back to [Constants](#)

SaveTouchstoneFileFormatReallImage

Description

The data format for the S-parameter saving - Real / Image.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared SaveTouchstoneFileFormatReallImage As Integer
```

C#

```
public static int SaveTouchstoneFileFormatReallImage
```

Visual C++

```
public:  
static int SaveTouchstoneFileFormatReallImage
```

JavaScript

```
CMT.Instruments.Attributes.saveTouchstoneFileFormatReallImage
```

Back to [Constants](#)

SaveTouchstoneFileS1p

Description

The Touchstone file type when saving S-parameters - S1P.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared SaveTouchstoneFileS1p As Integer
```

C#

```
public static int SaveTouchstoneFileS1p
```

Visual C++

```
public:  
static int SaveTouchstoneFileS1p
```

JavaScript

```
CMT.Instruments.Attributes.saveTouchstoneFileS1p
```

Back to [Constants](#)

SaveTouchstoneFileS2p

Description

The Touchstone file type when saving S-parameters - S2P.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared SaveTouchstoneFileS2p As Integer
```

C#

```
public static int SaveTouchstoneFileS2p
```

Visual C++

```
public:  
static int SaveTouchstoneFileS2p
```

JavaScript

```
CMT.Instruments.Attributes.saveTouchstoneFileS2p
```

Back to [Constants](#)

SaveTouchstoneFileS3p

Description

The Touchstone file type when saving S-parameters - S3P.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared SaveTouchstoneFileS3p As Integer
```

C#

```
public static int SaveTouchstoneFileS3p
```

Visual C++

```
public:  
static int SaveTouchstoneFileS3p
```

JavaScript

```
CMT.Instruments.Attributes.saveTouchstoneFileS3p
```

Back to [Constants](#)

SaveTouchstoneFileS4p

Description

The Touchstone file type when saving S-parameters - S4P.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared SaveTouchstoneFileS4p As Integer
```

C#

```
public static int SaveTouchstoneFileS4p
```

Visual C++

```
public:  
static int SaveTouchstoneFileS4p
```

JavaScript

```
CMT.Instruments.Attributes.saveTouchstoneFileS4p
```

Back to [Constants](#)

SaveTypeAll

Description

The type of the Analyzer or channel state - Measurement conditions, calibration, data and memory.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared SaveTypeAll As Integer
```

C#

```
public static int SaveTypeAll
```

Visual C++

```
public:  
static int SaveTypeAll
```

JavaScript

```
CMT.Instruments.Attributes.saveTypeAll
```

Back to [Constants](#)

SaveTypeState

Description

The type of the Analyzer or channel state - Measurement conditions.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared SaveTypeState As Integer
```

C#

```
public static int SaveTypeState
```

Visual C++

```
public:  
static int SaveTypeState
```

JavaScript

```
CMT.Instruments.Attributes.saveTypeState
```

Back to [Constants](#)

SaveTypeStateAndCal

Description

The type of the Analyzer or channel state - Measurement conditions and calibration.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared SaveTypeStateAndCal As Integer
```

C#

```
public static int SaveTypeStateAndCal
```

Visual C++

```
public:  
static int SaveTypeStateAndCal
```

JavaScript

```
CMT.Instruments.Attributes.saveTypeStateAndCal
```

Back to [Constants](#)

SaveTypeStateAndCalAndMem

Description

The type of the Analyzer or channel state - Measurement conditions, calibration and memory.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared SaveTypeStateAndCalAndMem As Integer
```

C#

```
public static int SaveTypeStateAndCalAndMem
```

Visual C++

```
public:  
static int SaveTypeStateAndCalAndMem
```

JavaScript

```
CMT.Instruments.Attributes.saveTypeStateAndCalAndMem
```

Back to [Constants](#)

SaveTypeStateAndTrace

Description

The type of the Analyzer or channel state - Measurement conditions and data.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared SaveTypeStateAndTrace As Integer
```

C#

```
public static int SaveTypeStateAndTrace
```

Visual C++

```
public:  
static int SaveTypeStateAndTrace
```

JavaScript

```
CMT.Instruments.Attributes.saveTypeStateAndTrace
```

Back to [Constants](#)

SegmentFrequencyOrder

Description

Frequency base (linear frequency axis).

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared SegmentFrequencyOrder As Integer
```

C#

```
public static int SegmentFrequencyOrder
```

Visual C++

```
public:  
static int SegmentFrequencyOrder
```

JavaScript

```
CMT.Instruments.Attributes.segmentFrequencyOrder
```

Back to [Constants](#)

SegmentIndexOrder

Description

Order base (linear axis of the point number).

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared SegmentIndexOrder As Integer
```

C#

```
public static int SegmentIndexOrder
```

Visual C++

```
public:  
static int SegmentIndexOrder
```

JavaScript

```
CMT.Instruments.Attributes.segmentIndexOrder
```

Back to [Constants](#)

StimulusTriggerStateHold

Description

The current state of the analyzer - Hold.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusTriggerStateHold As Integer
```

C#

```
public static int StimulusTriggerStateHold
```

Visual C++

```
public:  
static int StimulusTriggerStateHold
```

JavaScript

```
CMT.Instruments.Attributes.stimulusTriggerStateHold
```

Back to [Constants](#)

StimulusTriggerStateMeasure

Description

The current state of the analyzer - Measure (sweep in progress).

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusTriggerStateMeasure As Integer
```

C#

```
public static int StimulusTriggerStateMeasure
```

Visual C++

```
public:  
static int StimulusTriggerStateMeasure
```

JavaScript

```
CMT.Instruments.Attributes.stimulusTriggerStateMeasure
```

Back to [Constants](#)

StimulusTriggerStateWait

Description

The current state of the analyzer - Waiting for trigger.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared StimulusTriggerStateWait As Integer
```

C#

```
public static int StimulusTriggerStateWait
```

Visual C++

```
public:  
static int StimulusTriggerStateWait
```

JavaScript

```
CMT.Instruments.Attributes.stimulusTriggerStateWait
```

Back to [Constants](#)

SweepTypeLinFrequency

Description

Linear frequency sweep.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared SweepTypeLinFrequency As Integer
```

C#

```
public static int SweepTypeLinFrequency
```

Visual C++

```
public:  
static int SweepTypeLinFrequency
```

JavaScript

```
CMT.Instruments.Attributes.sweepTypeLinFrequency
```

Back to [Constants](#)

SweepTypeLogFrequency

Description

Logarithmic frequency sweep.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared SweepTypeLogFrequency As Integer
```

C#

```
public static int SweepTypeLogFrequency
```

Visual C++

```
public:  
static int SweepTypeLogFrequency
```

JavaScript

```
CMT.Instruments.Attributes.sweepTypeLogFrequency
```

Back to [Constants](#)

SweepTypePower

Description

Power sweep.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared SweepTypePower As Integer
```

C#

```
public static int SweepTypePower
```

Visual C++

```
public:  
static int SweepTypePower
```

JavaScript

```
CMT.Instruments.Attributes.sweepTypePower
```

Back to [Constants](#)

SweepTypeSegment

Description

Segment frequency sweep.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared SweepTypeSegment As Integer
```

C#

```
public static int SweepTypeSegment
```

Visual C++

```
public:  
static int SweepTypeSegment
```

JavaScript

```
CMT.Instruments.Attributes.sweepTypeSegment
```

Back to [Constants](#)

ThruAdditionMediaCoaxial

Description

The media of the thru - Specifies the coaxial.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ThruAdditionMediaCoaxial As Integer
```

C#

```
public static int ThruAdditionMediaCoaxial
```

Visual C++

```
public:  
static int ThruAdditionMediaCoaxial
```

JavaScript

```
CMT.Instruments.Attributes.thruAdditionMediaCoaxial
```

Back to [Constants](#)

ThruAdditionMediaWaveguide

Description

The media of the thru - Specifies the waveguide.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ThruAdditionMediaWaveguide As Integer
```

C#

```
public static int ThruAdditionMediaWaveguide
```

Visual C++

```
public:  
static int ThruAdditionMediaWaveguide
```

JavaScript

```
CMT.Instruments.Attributes.thruAdditionMediaWaveguide
```

Back to [Constants](#)

ThruAdditionUnitMeters

Description

The display units of the thru delay (length) - Selects meters.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ThruAdditionUnitMeters As Integer
```

C#

```
public static int ThruAdditionUnitMeters
```

Visual C++

```
public:  
static int ThruAdditionUnitMeters
```

JavaScript

```
CMT.Instruments.Attributes.thruAdditionUnitMeters
```

Back to [Constants](#)

ThruAdditionUnitSeconds

Description

The display units of the thru delay (length) - Selects seconds.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared ThruAdditionUnitSeconds As Integer
```

C#

```
public static int ThruAdditionUnitSeconds
```

Visual C++

```
public:  
static int ThruAdditionUnitSeconds
```

JavaScript

```
CMT.Instruments.Attributes.thruAdditionUnitSeconds
```

Back to [Constants](#)

TimeDomainGatingBandpass

Description

The gate type of the gating function - Bandpass.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainGatingBandpass As Integer
```

C#

```
public static int TimeDomainGatingBandpass
```

Visual C++

```
public:  
static int TimeDomainGatingBandpass
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainGatingBandpass
```

Back to [Constants](#)

TimeDomainGatingNotch

Description

The gate type of the gating function - Notch.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainGatingNotch As Integer
```

C#

```
public static int TimeDomainGatingNotch
```

Visual C++

```
public:  
static int TimeDomainGatingNotch
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainGatingNotch
```

Back to [Constants](#)

TimeDomainGatingShapeMaximum

Description

The gate shape of the gating function - Maximum.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainGatingShapeMaximum As Integer
```

C#

```
public static int TimeDomainGatingShapeMaximum
```

Visual C++

```
public:  
static int TimeDomainGatingShapeMaximum
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainGatingShapeMaximum
```

Back to [Constants](#)

TimeDomainGatingShapeMinimum

Description

The gate shape of the gating function - Minimum.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainGatingShapeMinimum As Integer
```

C#

```
public static int TimeDomainGatingShapeMinimum
```

Visual C++

```
public:  
static int TimeDomainGatingShapeMinimum
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainGatingShapeMinimum
```

Back to [Constants](#)

TimeDomainGatingShapeNormal

Description

The gate shape of the gating function - Normal.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainGatingShapeNormal As Integer
```

C#

```
public static int TimeDomainGatingShapeNormal
```

Visual C++

```
public:  
static int TimeDomainGatingShapeNormal
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainGatingShapeNormal
```

Back to [Constants](#)

TimeDomainGatingShapeWide

Description

The gate shape of the gating function - Wide.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainGatingShapeWide As Integer
```

C#

```
public static int TimeDomainGatingShapeWide
```

Visual C++

```
public:  
static int TimeDomainGatingShapeWide
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainGatingShapeWide
```

Back to [Constants](#)

TimeDomainReflectionOneWay

Description

The reflection distance for the time domain transformation - One Way.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainReflectionOneWay As Integer
```

C#

```
public static int TimeDomainReflectionOneWay
```

Visual C++

```
public:  
static int TimeDomainReflectionOneWay
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainReflectionOneWay
```

Back to [Constants](#)

TimeDomainReflectionRoundTrip

Description

The reflection distance for the time domain transformation - Round Trip.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainReflectionRoundTrip As Integer
```

C#

```
public static int TimeDomainReflectionRoundTrip
```

Visual C++

```
public:  
static int TimeDomainReflectionRoundTrip
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainReflectionRoundTrip
```

Back to [Constants](#)

TimeDomainTransformTypeBandpass

Description

The time domain transformation function = Bandpass

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainTransformTypeBandpass As Integer
```

C#

```
public static int TimeDomainTransformTypeBandpass
```

Visual C++

```
public:  
static int TimeDomainTransformTypeBandpass
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainTransformTypeBandpass
```

Back to [Constants](#)

TimeDomainTransformTypeLowpassImpulse

Description

The time domain transformation function = Lowpass Impulse.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainTransformTypeLowpassImpulse As Integer
```

C#

```
public static int TimeDomainTransformTypeLowpassImpulse
```

Visual C++

```
public:  
static int TimeDomainTransformTypeLowpassImpulse
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainTransformTypeLowpassImpulse
```

Back to [Constants](#)

TimeDomainTransformTypeLowpassStep

Description

The time domain transformation function = Lowpass Step.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainTransformTypeLowpassStep As Integer
```

C#

```
public static int TimeDomainTransformTypeLowpassStep
```

Visual C++

```
public:  
static int TimeDomainTransformTypeLowpassStep
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainTransformTypeLowpassStep
```

Back to [Constants](#)

TimeDomainUnitsFeet

Description

The transformation unit for the time domain - Feet.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainUnitsFeet As Integer
```

C#

```
public static int TimeDomainUnitsFeet
```

Visual C++

```
public:  
static int TimeDomainUnitsFeet
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainUnitsFeet
```

Back to [Constants](#)

TimeDomainUnitsMeters

Description

The transformation unit for the time domain - Meters.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainUnitsMeters As Integer
```

C#

```
public static int TimeDomainUnitsMeters
```

Visual C++

```
public:  
static int TimeDomainUnitsMeters
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainUnitsMeters
```

Back to [Constants](#)

TimeDomainUnitsSeconds

Description

The transformation unit for the time domain - Seconds.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainUnitsSeconds As Integer
```

C#

```
public static int TimeDomainUnitsSeconds
```

Visual C++

```
public:  
static int TimeDomainUnitsSeconds
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainUnitsSeconds
```

Back to [Constants](#)

TimeDomainWindowShapeArbitrary

Description

The time domain Kaiser–Bessel window shape - Arbitrary

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainWindowShapeArbitrary As Integer
```

C#

```
public static int TimeDomainWindowShapeArbitrary
```

Visual C++

```
public:  
static int TimeDomainWindowShapeArbitrary
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainWindowShapeArbitrary
```

Back to [Constants](#)

TimeDomainWindowShapeMaximum

Description

The time domain Kaiser–Bessel window shape - Maximum.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainWindowShapeMaximum As Integer
```

C#

```
public static int TimeDomainWindowShapeMaximum
```

Visual C++

```
public:  
static int TimeDomainWindowShapeMaximum
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainWindowShapeMaximum
```

Back to [Constants](#)

TimeDomainWindowShapeMinimum

Description

The time domain Kaiser–Bessel window shape - Minimum.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainWindowShapeMinimum As Integer
```

C#

```
public static int TimeDomainWindowShapeMinimum
```

Visual C++

```
public:  
static int TimeDomainWindowShapeMinimum
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainWindowShapeMinimum
```

Back to [Constants](#)

TimeDomainWindowShapeNormal

Description

The time domain Kaiser–Bessel window shape - Normal.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TimeDomainWindowShapeNormal As Integer
```

C#

```
public static int TimeDomainWindowShapeNormal
```

Visual C++

```
public:  
static int TimeDomainWindowShapeNormal
```

JavaScript

```
CMT.Instruments.Attributes.timeDomainWindowShapeNormal
```

Back to [Constants](#)

TraceDataMathAdd

Description

The math operation between the data trace and the memory trace - Addition Data + Mem.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TraceDataMathAdd As Integer
```

C#

```
public static int TraceDataMathAdd
```

Visual C++

```
public:  
static int TraceDataMathAdd
```

JavaScript

```
CMT.Instruments.Attributes.traceDataMathAdd
```

Back to [Constants](#)

TraceDataMathDiv

Description

The math operation between the data trace and the memory trace - Division Data / Mem.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TraceDataMathDiv As Integer
```

C#

```
public static int TraceDataMathDiv
```

Visual C++

```
public:  
static int TraceDataMathDiv
```

JavaScript

```
CMT.Instruments.Attributes.traceDataMathDiv
```

Back to [Constants](#)

TraceDataMathMult

Description

The math operation between the data trace and the memory trace - Multiplication Data x Mem.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TraceDataMathMult As Integer
```

C#

```
public static int TraceDataMathMult
```

Visual C++

```
public:  
static int TraceDataMathMult
```

JavaScript

```
CMT.Instruments.Attributes.traceDataMathMult
```

Back to [Constants](#)

TraceDataMathOff

Description

The math operation between the data trace and the memory trace - No math.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TraceDataMathOff As Integer
```

C#

```
public static int TraceDataMathOff
```

Visual C++

```
public:  
static int TraceDataMathOff
```

JavaScript

```
CMT.Instruments.Attributes.traceDataMathOff
```

Back to [Constants](#)

TraceDataMathSubt

Description

The math operation between the data trace and the memory trace - Subtraction Data – Mem.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TraceDataMathSubt As Integer
```

C#

```
public static int TraceDataMathSubt
```

Visual C++

```
public:  
static int TraceDataMathSubt
```

JavaScript

```
CMT.Instruments.Attributes.traceDataMathSubt
```

Back to [Constants](#)

TraceHoldMax

Description

The type of the trace hold function - Maximum hold.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TraceHoldMax As Integer
```

C#

```
public static int TraceHoldMax
```

Visual C++

```
public:  
static int TraceHoldMax
```

JavaScript

```
CMT.Instruments.Attributes.traceHoldMax
```

Back to [Constants](#)

TraceHoldMin

Description

The type of the trace hold function - Minimum hold.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TraceHoldMin As Integer
```

C#

```
public static int TraceHoldMin
```

Visual C++

```
public:  
static int TraceHoldMin
```

JavaScript

```
CMT.Instruments.Attributes.traceHoldMin
```

Back to [Constants](#)

TraceHoldOff

Description

The type of the trace hold function - Turns off the trace hold function.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TraceHoldOff As Integer
```

C#

```
public static int TraceHoldOff
```

Visual C++

```
public:  
static int TraceHoldOff
```

JavaScript

```
CMT.Instruments.Attributes.traceHoldOff
```

Back to [Constants](#)

TriggerModeContinuous

Description

A sweep actuation occurs every time a trigger signal is detected.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TriggerModeContinuous As Integer
```

C#

```
public static int TriggerModeContinuous
```

Visual C++

```
public:  
static int TriggerModeContinuous
```

JavaScript

```
CMT.Instruments.Attributes.triggerModeContinuous
```

Back to [Constants](#)

TriggerModeHold

Description

Sweep actuation is off in the channel, trigger signals do not affect the channel.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TriggerModeHold As Integer
```

C#

```
public static int TriggerModeHold
```

Visual C++

```
public:  
static int TriggerModeHold
```

JavaScript

```
CMT.Instruments.Attributes.triggerModeHold
```

Back to [Constants](#)

TriggerModeSingle

Description

One sweep actuation occurs with trigger signal detection after the mode has been enabled; after the sweep is complete the channel modes changes to hold.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TriggerModeSingle As Integer
```

C#

```
public static int TriggerModeSingle
```

Visual C++

```
public:  
static int TriggerModeSingle
```

JavaScript

```
CMT.Instruments.Attributes.triggerModeSingle
```

Back to [Constants](#)

TriggerOutputFunctionAsampling

Description

The trigger output function - After sampling pulse.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TriggerOutputFunctionAsampling As Integer
```

C#

```
public static int TriggerOutputFunctionAsampling
```

Visual C++

```
public:  
static int TriggerOutputFunctionAsampling
```

JavaScript

```
CMT.Instruments.Attributes.triggerOutputFunctionAsampling
```

Back to [Constants](#)

TriggerOutputFunctionBsampling

Description

The trigger output function - Before sampling pulse.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TriggerOutputFunctionBsampling As Integer
```

C#

```
public static int TriggerOutputFunctionBsampling
```

Visual C++

```
public:  
static int TriggerOutputFunctionBsampling
```

JavaScript

```
CMT.Instruments.Attributes.triggerOutputFunctionBsampling
```

Back to [Constants](#)

TriggerOutputFunctionBsetup

Description

The trigger output function - Before frequency setup pulse.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TriggerOutputFunctionBsetup As Integer
```

C#

```
public static int TriggerOutputFunctionBsetup
```

Visual C++

```
public:  
static int TriggerOutputFunctionBsetup
```

JavaScript

```
CMT.Instruments.Attributes.triggerOutputFunctionBsetup
```

Back to [Constants](#)

TriggerOutputFunctionMeasurement

Description

The trigger output function - Measurement sweep signal.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TriggerOutputFunctionMeasurement As Integer
```

C#

```
public static int TriggerOutputFunctionMeasurement
```

Visual C++

```
public:  
static int TriggerOutputFunctionMeasurement
```

JavaScript

```
CMT.Instruments.Attributes.triggerOutputFunctionMeasurement
```

Back to [Constants](#)

TriggerOutputFunctionReadyForTrigger

Description

The trigger output function - Ready for trigger signal.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TriggerOutputFunctionReadyForTrigger As Integer
```

C#

```
public static int TriggerOutputFunctionReadyForTrigger
```

Visual C++

```
public:  
static int TriggerOutputFunctionReadyForTrigger
```

JavaScript

```
CMT.Instruments.Attributes.triggerOutputFunctionReadyForTrigger
```

Back to [Constants](#)

TriggerOutputFunctionSweepEnd

Description

The trigger output function - End of sweep pulse.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TriggerOutputFunctionSweepEnd As Integer
```

C#

```
public static int TriggerOutputFunctionSweepEnd
```

Visual C++

```
public:  
static int TriggerOutputFunctionSweepEnd
```

JavaScript

```
CMT.Instruments.Attributes.triggerOutputFunctionSweepEnd
```

Back to [Constants](#)

TriggerOutputPolarityNegative

Description

The polarity of the trigger output - Negative edge.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TriggerOutputPolarityNegative As Integer
```

C#

```
public static int TriggerOutputPolarityNegative
```

Visual C++

```
public:  
static int TriggerOutputPolarityNegative
```

JavaScript

```
CMT.Instruments.Attributes.triggerOutputPolarityNegative
```

Back to [Constants](#)

TriggerOutputPolarityPositive

Description

The polarity of the trigger output - Positive edge.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TriggerOutputPolarityPositive As Integer
```

C#

```
public static int TriggerOutputPolarityPositive
```

Visual C++

```
public:  
static int TriggerOutputPolarityPositive
```

JavaScript

```
CMT.Instruments.Attributes.triggerOutputPolarityPositive
```

Back to [Constants](#)

TriggerRoutePxiTrig0

Description

The connector to use for the external trigger input - Backplane Trigger Line (PXI TRIG0).

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TriggerRoutePxiTrig0 As Integer
```

C#

```
public static int TriggerRoutePxiTrig0
```

Visual C++

```
public:  
static int TriggerRoutePxiTrig0
```

JavaScript

```
CMT.Instruments.Attributes.triggerRoutePxiTrig0
```

Back to [Constants](#)

TriggerRoutePxiTrig1

Description

The connector to use for the external trigger input - Backplane Trigger Line (PXI TRIG1).

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TriggerRoutePxiTrig1 As Integer
```

C#

```
public static int TriggerRoutePxiTrig1
```

Visual C++

```
public:  
static int TriggerRoutePxiTrig1
```

JavaScript

```
CMT.Instruments.Attributes.triggerRoutePxiTrig1
```

Back to [Constants](#)

TriggerRoutePxiTrig2

Description

The connector to use for the external trigger input - Backplane Trigger Line (PXI TRIG2).

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TriggerRoutePxiTrig2 As Integer
```

C#

```
public static int TriggerRoutePxiTrig2
```

Visual C++

```
public:  
static int TriggerRoutePxiTrig2
```

JavaScript

```
CMT.Instruments.Attributes.triggerRoutePxiTrig2
```

Back to [Constants](#)

TriggerRoutePxiTrig3

Description

The connector to use for the external trigger input - Backplane Trigger Line (PXI TRIG3).

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TriggerRoutePxiTrig3 As Integer
```

C#

```
public static int TriggerRoutePxiTrig3
```

Visual C++

```
public:  
static int TriggerRoutePxiTrig3
```

JavaScript

```
CMT.Instruments.Attributes.triggerRoutePxiTrig3
```

Back to [Constants](#)

TriggerRoutePxiTrig4

Description

The connector to use for the external trigger input - Backplane Trigger Line (PXI TRIG4).

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TriggerRoutePxiTrig4 As Integer
```

C#

```
public static int TriggerRoutePxiTrig4
```

Visual C++

```
public:  
static int TriggerRoutePxiTrig4
```

JavaScript

```
CMT.Instruments.Attributes.triggerRoutePxiTrig4
```

Back to [Constants](#)

TriggerRoutePxiTrig5

Description

The connector to use for the external trigger input - Backplane Trigger Line (PXI TRIG5).

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TriggerRoutePxiTrig5 As Integer
```

C#

```
public static int TriggerRoutePxiTrig5
```

Visual C++

```
public:  
static int TriggerRoutePxiTrig5
```

JavaScript

```
CMT.Instruments.Attributes.triggerRoutePxiTrig5
```

Back to [Constants](#)

TriggerRoutePxiTrig6

Description

The connector to use for the external trigger input - Backplane Trigger Line (PXI TRIG6).

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TriggerRoutePxiTrig6 As Integer
```

C#

```
public static int TriggerRoutePxiTrig6
```

Visual C++

```
public:  
static int TriggerRoutePxiTrig6
```

JavaScript

```
CMT.Instruments.Attributes.triggerRoutePxiTrig6
```

Back to [Constants](#)

TriggerRoutePxiTrig7

Description

The connector to use for the external trigger input - Backplane Trigger Line (PXI TRIG7).

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TriggerRoutePxiTrig7 As Integer
```

C#

```
public static int TriggerRoutePxiTrig7
```

Visual C++

```
public:  
static int TriggerRoutePxiTrig7
```

JavaScript

```
CMT.Instruments.Attributes.triggerRoutePxiTrig7
```

Back to [Constants](#)

TriggerRouteSmb

Description

The connector to use for the external trigger input - Front panel connector "Ext Trig In".

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TriggerRouteSmb As Integer
```

C#

```
public static int TriggerRouteSmb
```

Visual C++

```
public:  
static int TriggerRouteSmb
```

JavaScript

```
CMT.Instruments.Attributes.triggerRouteSmb
```

Back to [Constants](#)

TriggerRouteStar

Description

The connector to use for the external trigger input - Backplane Trigger Line (PXI STAR).

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TriggerRouteStar As Integer
```

C#

```
public static int TriggerRouteStar
```

Visual C++

```
public:  
static int TriggerRouteStar
```

JavaScript

```
CMT.Instruments.Attributes.triggerRouteStar
```

Back to [Constants](#)

TriggerScopeActiveChannel

Description

The trigger scope - Active channel.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TriggerScopeActiveChannel As Integer
```

C#

```
public static int TriggerScopeActiveChannel
```

Visual C++

```
public:  
static int TriggerScopeActiveChannel
```

JavaScript

```
CMT.Instruments.Attributes.triggerScopeActiveChannel
```

Back to [Constants](#)

TriggerScopeAllChannel

Description

The trigger scope - All channels.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TriggerScopeAllChannel As Integer
```

C#

```
public static int TriggerScopeAllChannel
```

Visual C++

```
public:  
static int TriggerScopeAllChannel
```

JavaScript

```
CMT.Instruments.Attributes.triggerScopeAllChannel
```

Back to [Constants](#)

TriggerSourceBus

Description

The trigger signal is generated by a command communicated from an external computer from a program controlling the Analyzer via automation.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TriggerSourceBus As Integer
```

C#

```
public static int TriggerSourceBus
```

Visual C++

```
public:  
static int TriggerSourceBus
```

JavaScript

```
CMT.Instruments.Attributes.triggerSourceBus
```

Back to [Constants](#)

TriggerSourceExternal

Description

The external trigger input is used as a trigger signal source.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TriggerSourceExternal As Integer
```

C#

```
public static int TriggerSourceExternal
```

Visual C++

```
public:  
static int TriggerSourceExternal
```

JavaScript

```
CMT.Instruments.Attributes.triggerSourceExternal
```

Back to [Constants](#)

TriggerSourceInternal

Description

The next trigger signal is generated by the Analyzer on completion of each sweep.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TriggerSourceInternal As Integer
```

C#

```
public static int TriggerSourceInternal
```

Visual C++

```
public:  
static int TriggerSourceInternal
```

JavaScript

```
CMT.Instruments.Attributes.triggerSourceInternal
```

Back to [Constants](#)

TriggerSourceManual

Description

The trigger signal is generated by pressing the corresponding softkey.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared TriggerSourceManual As Integer
```

C#

```
public static int TriggerSourceManual
```

Visual C++

```
public:  
static int TriggerSourceManual
```

JavaScript

```
CMT.Instruments.Attributes.triggerSourceManual
```

Back to [Constants](#)

UploadTouchstoneFileToActiveTraceMemory

Description

Loads the Touchstone file to the memory trace.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared UploadTouchstoneFileToActiveTraceMemory As Integer
```

C#

```
public static int UploadTouchstoneFileToActiveTraceMemory
```

Visual C++

```
public:  
static int UploadTouchstoneFileToActiveTraceMemory
```

JavaScript

```
CMT.Instruments.Attributes.uploadTouchstoneFileToActiveTraceMemory
```

Back to [Constants](#)

UploadTouchstoneFileToSParameters

Description

Loads the Touchstone file to S-parameters.

Namespace: [CMT.Instruments](#)

Assembly: CmtNA.Fx40 (in CmtNA.Fx40.dll)

Visual Basic (Declaration)

```
Public Shared UploadTouchstoneFileToSParameters As Integer
```

C#

```
public static int UploadTouchstoneFileToSParameters
```

Visual C++

```
public:  
static int UploadTouchstoneFileToSParameters
```

JavaScript

```
CMT.Instruments.Attributes.uploadTouchstoneFileToSParameters
```

Back to [Constants](#)

Installation

Location of Files

The CMT VNA VI.NET driver installer deploys and registers a variety of files on the target computer. The notation <viInstallDir> is used to denote the directory in which the VI.NET Shared Components have been installed.

By default, <viInstallDir> for x86 is set C:\Program Files (x86)\VI Foundation\VMMicrosoft.NET.

By default, <viInstallDir> for x64 is set C:\Program Files\VI Foundation\VMMicrosoft.NET.

Substitute your install directory if the VI Shared Components were installed in an alternate location.

Platform	File	Description
x86	<viInstallDir>\Framework32\v4.0.30319\CmtNA VI.NET Driver 20.3.1\CmtNA.Fx40.dll	Driver assembly.
	<viInstallDir>\Framework32\v4.0.30319\CmtNA VI.NET Driver 20.3.1\CmtNA.Fx40.xml	Driver XML help.
	<viInstallDir>\Framework32\v4.0.30319\CmtNA VI.NET Driver 20.3.1\Help\CmtNA_VI_NET_Driver_Reference. chm	CHM help file.
	<viInstallDir>\Framework32\v4.0.30319\CmtNA VI.NET Driver 20.3.1\Examples\	Driver example programs.
	<viInstallDir>\Framework32\v4.0.30319\CmtNA VI.NET Driver 20.3.1\Source\	Optionally installed driver source code project.
x64	<viInstallDir>\Framework64\v4.0.30319\CmtNA VI.NET Driver 20.3.1\CmtNA.Fx40.dll	Driver assembly.
	<viInstallDir>\Framework64\v4.0.30319\CmtNA VI.NET Driver 20.3.1\CmtNA.Fx40.xml	Driver XML help.

Platform	File	Description
	<IvInstallDir>\Framework64\v4.0.30319\CmtNA IVI.NET Driver 20.3.1\Help\CmtNA_IVI_NET_Driver_Reference. chm	CHM help file.
	<IvInstallDir>\Framework64\v4.0.30319\CmtNA IVI.NET Driver 20.3.1\Examples\	Driver example programs.
	<IvInstallDir>\Framework64\v4.0.30319\CmtNA IVI.NET Driver 20.3.1\Source\	Optionally installed driver source code project.

IVI Shared Components and IVI.NET Shared Components

Before any IVI driver is installed on a system, the IVI Shared Components and the IVI.NET Shared Components must both be properly installed.

For additional information on the IVI Shared Components and the IVI.NET Shared Components, visit the [IVI Foundation web site](#).

To determine which version of the IVI Shared Components is installed on your system:

1. Navigate to <IvInstallDir>\Bin. Typically this is:
 - C:\Program Files (x86)\IVI Foundation\IVI\Bin for x86
 - C:\Program Files\IVI Foundation\IVI\Bin for x64
2. Locate the file IvSharedComponentVersion.dll
3. Right-click on that file and choose **Properties**
4. In the Properties Window, click the **Version** tab
5. The **File version** field indicates which version of the IVI Shared Components are installed

To determine which version of the IVI.NET Shared Components are installed on your system:

1. Navigate to:
 - <IvInstallDir>\Microsoft.NET\Framework32\v2.0.50727\IvFoundationShared Components X.X for x86

- <IvInstallDir>\Microsoft.NET\Framework64\v2.0.50727\MiFoundationShared Components X.X for x64

(where X.X is the major.minor version number).

2. Locate the file MiFoundationSharedComponentsVersion.dll.
3. Right-click on that file and choose **Properties**.
4. In the Properties Window, click the **Version** tab.
5. The **File version** field indicates which version of the .NET Shared Components are installed.

CmtNA IVI-C Driver Components

Before working with the CMT VNA .NET driver, the CmtNA IVI-C Driver must be installed. For more detailed information about installation see the CmtNA IVI-C Driver documentation.

Uninstallation

The only proper way to explicitly remove the driver from the system is to use the uninstallation program. Simply removing the installed files will not properly uninstall the driver.

To uninstall the driver:

1. From the Windows **Start** menu, choose **All Programs** and **Select Panel**.
2. Double-click **Add** or **Remove Programs**.
3. Select the driver and click **Remove**.

Copyright

Under the copyright laws, this publication must not be reproduced or transmitted in any form, electronic or mechanical, including photocopying, recording, storing in an information retrieval system, or translating, in whole or in part, without the prior written consent of Copper Mountain Technologies.

Copper Mountain Technologies respects the intellectual property of others, and we ask our users to do the same. CMT software is protected by copyright and other intellectual property laws. Where CMT software may be used to reproduce software or other materials belonging to others, you may use CMT software only to reproduce materials that you may reproduce in accordance with the terms of any applicable license or other legal restriction.