THOMSON REUTERS EIKON

EIKON FOR DEVELOPERS

March 09, 2016

THOMSON REUTERS™

Version published on March 09, 2016.

Updates in this version:

- "RHistory API " on page 155

Contents

AdxBasket
AdxBond
AdxCalcMethod
AdxCapFloor
AdxCashFlow
AdxCDOTranche
AdxChooser
AdxConvBond
AdxCorrelation
AdxCrossCurrency
AdxCurrency
AdxDefault
AdxDigitalCapFloor
AdxDividendModel
AdxFixedLeg
AdxFloatLeg
AdxForex
AdxFra
AdxFrn
AdxFrq
AdxFuture
AdxFxModel
AdxIlb
AdxInit
AdxLibor
AdxMapLibor
AdxModelBuilder
AdxNToDefaultCDS
AdxOpBinary
AdxOpLookBack
AdxOption
AdxParse
AdxRainbow
AdxRateModel
AdxRepo
AdxRiskModel
AdxSchedule

AdxDcbType
AdxDCP
AdxDilutionFlag
AdxDividendCstGrowthType
AdxDividendType
AdxDivUserDefined
AdxEndOfMonthConvention
AdxErrorMode
AdxExerciseMode
AdxExerciseStrategy
AdxFrequency
AdxFrequencyType
AdxFrom
AdxFTType
AdxFutureReferenceRule
AdxIC
AdxICF
AdxICM
AdxIndexDate
AdxInstrumentType
AdxInterceptYesNo
AdxIOType
AdxIrsPvbpMethod
AdxLayOut
AdxLegAttr
AdxLookBackType
AdxNormalType
AdxOriginPeriod
AdxPayment
AdxPEX
AdxPxType
AdxQuotationMode
AdxRateModelType
AdxRateOfReturn
AdxRateType
AdxReferenceDate
AdxReset

VBA Programming in Thomson Reuters Eikon Excel

You can use VBA to customize the application environment to your business needs. To start using VBA in Thomson Reuters Eikon Excel, you must perform the additional setup steps described below.

How to set up VBA in Thomson Reuters Eikon – Microsoft Office

1 Open Microsoft Excel (or Word or PowerPoint).

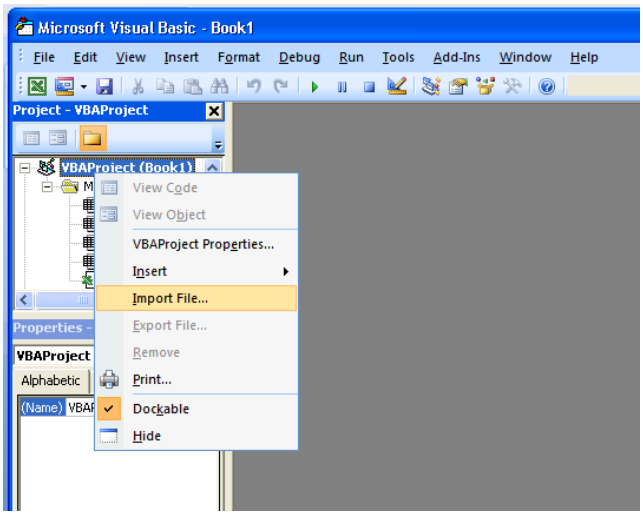
2 Click *Developer* > *Visual Basic* OR press *Alt+F11*

The VBA screen opens.

ⓘ The Developer tab is hidden in Microsoft Excel 2007. To display this tab, see "How to display the Developer tab" on page 149.

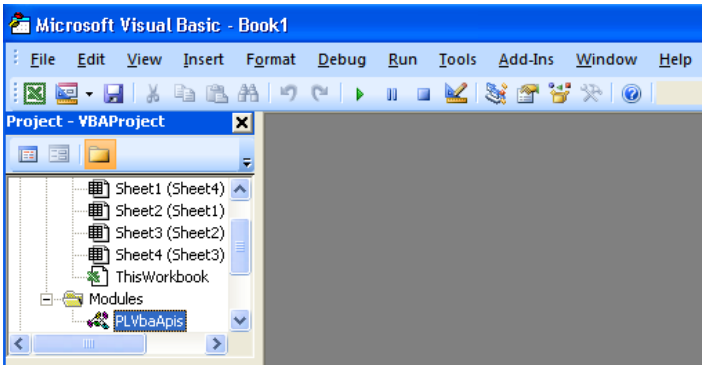
3 Right-click *VBAProject (Bookn)* and choose *Import File*.

The *Import File* window opens.



4 Select *PLVbaApis.bas* from the Thomson Reuters Eikon installation folder and click *Open*.

The module is added to the VBA project.



5 In the VBA project, click *Tools* > *References* to add the references to Thomson Reuters Eikon APIs.

- RTX.DLL
- ADXOO.DLL

- ADXFO.DLL
- RSEARCH.DLL
- DEX2.DLL

Note

If they do not appear by default, browse to `C:\Program Files (x86)\Thomson Reuters\Eikon\X\Bin\`

And use the following folder for DEX2: `C:\Program Files (x86)\Thomson Reuters\Eikon\X\Bin\Apps\TR.OFFICE.CORE\0.0.0.0\Bin\`

Working with Public APIs

Introduction

Public APIs allow you to access the data and functionality of Thomson Reuters Eikon from customized applications. They are registered in Reuters 3000 Xtra 5.x. To use them in Thomson Reuters Eikon, you must activate the optional Thomson Reuters Eikon Public API. See Introducing API Migration.

Public APIs in Thomson Reuters Eikon

Public APIs available in Thomson Reuters Eikon include:

API Migration

You can refer to these topics for more information on Public API migration:

- API Migration
- Using AdfinX-Analytics in Thomson Reuters Eikon
- Using AdfinX-Realtime in Thomson Reuters Eikon
- API Comparison Between Reuters 3000 Xtra and Thomson Reuters Eikon

Thomson Reuters API Usage in VBA

AdfinX Analytics 2.0 Function Library

The AdfinX Analytics Function Library allows you to access Adfin function APIs.

How to add a reference to AdfinX functions

You must add a reference to AdfinX functions to make it available to user application codes.

1 Open the *Visual Basic Editor*.

2 Choose *Tools* > *References* > *Browse*.

3 Locate the `Adxfo.dll` file under the `Bin` folder of Thomson Reuters Eikon

By default, the folder is `C:\Program Files (x86)\Thomson Reuters\Eikon\X\Bin\`

4 Click *Open*.

List of AdfinX functions

AdxBondModule

The methods in `AdxBondModule` cover:

- coupon related functions (number, date, and accrued interest)
- main yield and price functions
- bond derivatives (duration, volatility, convexity, PVBP)
- spread (over a benchmark, over a risk free curve), option adjusted spread

AdfinX Functions in AdxBondModule

ⓘ To find out more about these functions, consult the Adfin Library online help. You may be prompted to sign in with your Eikon user ID and password.

Accrued	AdBondDeriv	AdBondECSDeriv
AdBondECSPrice	AdBondECSSpread	AdBondHedgesRatios
AdBondPrice	AdBondProceeds	AdBondReturn
AdBondSpread	AdBondSpreadMarket	AdBondYield
AdCmsFrnCashFlows	AdCmsFrnDeriv	AdCmsFrnPrice
AdCmsFrnYield	AdCmsSpreadFrnCashFlows	AdCmsSpreadFrnDeriv
AdCmsSpreadFrnPrice	AdCmsSpreadFrnYield	AdCmtDeriv
AdCmtPrice	AdFrnCalcCpn	AdFrnCashFlows
AdFrnDeriv	AdFrnMargin	AdFrnPrice
AdFrnRepo	AdFrnYield	AdIlbDeriv
AdIlbRepo	AdRangeAccrualNoteCashFlows	AdRangeAccrualNoteDeriv
AdRangeAccrualNotePrice	AdRangeAccrualNoteYield	BdCalcCpn
BdCashFlows	BdConvFactor	BdCpnCrv
BdCpnValue	BdIrsStructure	BdPvbpCrv
BdRepo	BdSettle	BdSettleLock
CfAvgLife	CfConv	CfDur
CfPvbp	CfPx	CfPxCrv
CfRepo	CfVol	CfYld
CpnNext	CpnNumber	CpnPrev
IlbCalcCpn	IlbCashFlows	IlbPx
IlbYld		

ErrorCode Property

The `ErrorCode` property of `AdxBondModule` stores the last error code generated by the object.

Syntax

```
ErrorCode() As Long
```

VBA sample

Example	Explanation
`Dim bd as AdxBondModule`	Declares an object of the type AdxBondModule
`Set bd = New AdxBondModule`	Creates an AdxBondModule object
`If bd.ErrorCode <> 0 Then`	bd.ErrorCode contains an error
`MsgBox (bd.ErrorCode)`	Displays the error code in a message box
`End If`	

Arguments

None

ErrorMode Property

The `ErrorMode` property of `AdxBondModule` sets the error mode.

Modes

Keyword	In this mode
`DIALOGBOX`	a dialog box is displayed when an error occurs
`EXCEPTION`	an exception is thrown when an error occurs
`NO_EXCEPTION`	the user must call the property ErrorCode to know the error encountered

Syntax

```
ErrorMode() As AdxErrorMode
```

VBA sample

Example	Explanation
`Dim bd as AdxBondModule`	Declares an object of the type AdxBondModule
`Set bd = New AdxBondModule`	Creates an AdxBondModule object
`bd.ErrorMode = NO_EXCEPTION`	bd.ErrorMode is set to `NO_EXCEPTION`

Arguments

None

Default value

By default the error mode is set to `EXCEPTION`.

ErrorString Property

The `ErrorString` property of `AdxBondModule` stores the last error code generated by the object.

Syntax

```
ErrorString() As String
```

VBA sample

Example	Explanation
`Dim bd as AdxBondModule`	Declares an object of variable the type AdxBondModule
`Set bd = New AdxBondModule`	Creates an AdxBondModule object
`bd.ErrorMode = EXCEPTION`	bd.ErrorMode must be set to `EXCEPTION` to use the ErrorString
`On Error Go To Handle`	Error handler executes code after Handle when an error occurs
`RES = bd.BdSettle ("24-JAN-00", "GLT")`	The style 'GLT' does not exist. The function BdSettle returns an error
`Exit Sub`	Exits sub or function before the handle or code is executed
`Handle MsgBox (bd.ErrorString)`	Displays the ErrorString in a message box. In this case, the error string is ('Error#7008 – SetStructure with an unrecognized keyword.')

Arguments

None

AdxCommodityModule

You can use `AdxCommodityModule` functions to price convertible bond instruments.

AdfinX Functions in AdxCommodityModule

To find out more about these functions, consult the Adfin Library online help. You may be prompted to sign in with your Eikon user ID and password.

`AdceCalcCalendarStrips` `AdceCalcContractStrips` `AdceCalcOutrights` `AdceForwardTermStructure`

ErrorCode Property

The `ErrorCode` property of `AdxBondModule` stores the last error code generated by the object.

Syntax

```
ErrorCode() As Long
```

VBA sample

Example	Explanation
`Dim bd as AdxBondModule`	Declares an object of the type AdxBondModule
`Set bd = New AdxBondModule`	Creates an AdxBondModule object
`If bd.ErrorCode <> 0 Then`	bd.ErrorCode contains an error
`MsgBox (bd.ErrorCode)`	Displays the error code in a message box
`End If`	

Arguments

None

ErrorMode Property

The `ErrorMode` property of `AdxBondModule` sets the error mode.

Modes

Keyword	In this mode
DIALOGBOX	a dialog box is displayed when an error occurs
EXCEPTION	an exception is thrown when an error occurs
NO_EXCEPTION	the user must call the property ErrorCode to know the error encountered

Syntax

```
ErrorMode() As AdxErrorMode
```

VBA sample

Example	Explanation
Dim bd as AdxBondModule	Declares an object of the type AdxBondModule
Set bd = New AdxBondModule	Creates an AdxBondModule object
bd.ErrorMode = NO_EXCEPTION	bd.ErrorMode is set to NO_EXCEPTION

Arguments

None

Default value

By default the error mode is set to EXCEPTION.

ErrorString Property

The ErrorString property of AdxBondModule stores the last error code generated by the object.

Syntax

```
ErrorString() As String
```

VBA sample

Example	Explanation
`Dim bd as AdxBondModule`	Declares an object of variable the type AdxBondModule
`Set bd = New AdxBondModule`	Creates an AdxBondModule object
`bd.ErrorMode = EXCEPTION`	bd.ErrorMode must be set to `EXCEPTION` to use the ErrorString
`On Error Go To Handle`	Error handler executes code after Handle when an error occurs
`RES = bd.BdSettle ("24-JAN-00", "GLT")`	The style 'GLT' does not exist. The function BdSettle returns an error
`Exit Sub`	Exits sub or function before the handle or code is executed
`Handle MsgBox (bd.ErrorString)`	Displays the ErrorString in a message box. In this case, the error string is ('Error#7008 – SetStructure with an unrecognized keyword.')

Arguments

None

AdxConvBondModule

You can use `AdxConvBondModule` functions to to price convertible bond instruments.

AdfinX Functions in AdxConvBondModule

To find out more about these functions, consult the Adfin Library online help. You may be prompted to sign in with your Eikon user ID and password.

`AdConvBdDeriv`	`AdConvCalcCpn`	`AdConvCashFlows`
`AdConvImpliedVol`	`AdConvOpDeriv`	`AdConvPrice`
`AdConvRatios`	`AdConvSpread`	`AdConvYield`

ErrorCode Property

The `ErrorCode` property of `AdxConvBondModule` stores the last error code generated by the object.

Syntax

```
ErrorCode() As Long
```

VBA sample

Example	Explanation
`Dim cd as AdxConvBondModule`	Declares an object of the type AdxConvBondModule
`Set cd = New AdxConvBondModule`	Creates an AdxConvBondModule object
`If cd.ErrorCode <> 0 Then`	cd.ErrorCode contains an error
`MsgBox (cd.ErrorCode)`	Displays the error code in a message box
`End If`	

Arguments

None

ErrorMode Property

The `ErrorMode` property of `AdxConvBondModule` sets the error mode.

Modes

Keyword	In this mode
DIALOGBOX	a dialog box is displayed when an error occurs
EXCEPTION	an exception is thrown when an error occurs
NO_EXCEPTION	the user must call the property ErrorCode to know the error encountered

Syntax

```
ErrorMode() As AdxErrorMode
```

VBA sample

Example	Explanation
Dim cd as AdxConvBondModule	Declares an object of the type AdxConvBondModule
Set cd = New AdxConvBondModule	Creates an AdxConvBondModule object
cd.ErrorMode = NO_EXCEPTION	cd.ErrorMode is set to NO_EXCEPTION

Arguments

None

Default value

By default the error mode is set to EXCEPTION.

ErrorString Property

The ErrorString property of AdxConvBondModule stores the last error code generated by the object.

Syntax

```
ErrorString() As String
```

VBA sample

Example	Explanation
`Dim cd as AdxConvBondModule`	Declares an object of the type AdxConvBondModule
`Set cd = New AdxConvBondModule`	Creates an AdxConvBondModule object
`cd.ErrorMode = EXCEPTION`	cd.ErrorMode must be set to `EXCEPTION` to use the ErrorString
`On Error Go To Handle`	Error handler executes code after Handle when an error occurs
`RES = cd.AdConvCalcCpn("24-APR-01", "25-FEB-01", "0.08", "CONVSRATIO:05OCT00:01JAN04:1.12", "LAY:V")`	The keyword `CONVSRATIO` does not exist. The function AdConvCalcCpn returns an error
`Exit Sub` `Handle`	Exits sub or function before the handle or code is executed
`MsgBox(cd.ErrorString)`	Displays the ErrorString in a message box. In this case, the error string contains 'Error#7008 – SetStructure with an unrecognized keyword.'

Arguments

None

AdxCreditModule

You can use `AdxCreditModule` functions to calculate:

- valuation
- pricing
- credit modelling and calibration
- probability curve building
- credit zero-coupon curve building

AdfinX Functions in AdxCreditModule

To find out more about these functions, consult the Adfin Library online help. You may be prompted to sign in with your Eikon user ID and password.

`AdCdoCorrelation`
`AdCdoNpv`
`AdCdoSpread`
`AdCdsDeriv`
`AdCdsFeeLegCashFlow`
`AdCdsNpv`
`AdCdsSpread`
`AdCreditBondPrice`
`AdCreditFrnPrice`
`AdCreditStructure`
`AdCreditZcCurve`
`AdDefaultProba`
`AdFxCdsNpv`
`AdFxCdsSpread`
`AdJLTCreditStructure`
`AdNToDefaultCdsNpv`
`AdNToDefaultCdsSpread`
`AdTrsNpv`
`AdTrsSpread`

ErrorCode Property

The `ErrorCode` property of `AdxCreditModule` stores the last error code generated by the object.

Syntax

```
ErrorCode() As Long
```

VBA sample

Example	Explanation
`Dim ct as AdxCreditModule`	Declares an object of the type AdxCreditModule
`Set ct = New AdxCreditModule`	Creates an AdxCreditModule object
`If ct.ErrorCode <> 0 Then`	ct.ErrorCode contains an error
`MsgBox (ct.ErrorCode)`	Displays the error code in a message box
`End If`	

Arguments

None

ErrorMode Property

The `ErrorMode` property of `AdxCreditModule` sets the error mode.

Modes

Keyword	In this mode
`DIALOGBOX`	a dialog box is displayed when an error occurs
`EXCEPTION`	an exception is thrown when an error occurs
`NO_EXCEPTION`	the user must call the property ErrorCode to know the error encountered

Syntax

```
ErrorMode() As AdxErrorMode
```

VBA sample

Example	Explanation
`Dim ct as AdxCreditModule`	Declares an object of type AdxCreditModule
`Set ct = New AdxCreditModule`	Creates an AdxCreditModule object
`ct.ErrorMode = NO_EXCEPTION`	ct.ErrorMode is set to `NO_EXCEPTION`

Arguments

None

Default value

By default the error mode is set to `EXCEPTION`.

ErrorString Property

The `ErrorString` property of `AdxCreditModule` retrieves the last error code generated by the object.

Syntax

```
ErrorString() As String
```

VBA sample

Example	Explanation
`Dim ct as AdxCreditModule`	Declares an object of the the type AdxCreditModule
`Set ct = New AdxCreditModule`	Creates an AdxCreditModule object
`ct.ErrorMode = EXCEPTION`	ex.ErrorMode must be set to `EXCEPTION` to use the ErrorString
`On Error Go To Handle`	Error handler executes code after Handle when an error occurs
`Res = ct.AdCdsNpv("19SEP02", "15SEP02", "5Y", 200, 0.3, [ratea].Value, [credita].Value, "RATETYPE:CMP CDSTYPE:AMERCDS CLDR:EMU_FI LFLOAT AOD:YES LFIXED FRQ:4 CCM:MMA0", "RISKMODEL:CURVE RECOVERY:0.3 NBDAYS:5 ND:DIS", "RM:YC ZCTYPE:DF IMCUBD")`	The RateStructure is incorrect. The function `AdCdsNpv` returns an error
`Exit Sub` `Handle`	Exits sub or function before the handle or code is executed
`MsgBox(ct.ErrorString)`	Displays the ErrorString in a message box. In this case, the error string contains 'Error#7601 – RateStructure: invalid keyword.'

Arguments

None

AdxDateModule

You can use `AdxDateModule` functions to manage dates. These functions calculate settlement dates, taking into account holidays and other related factors.

AdfinX Functions in AdxDateModule

To find out more about these functions, consult the Adfin Library online help. You may be prompted to sign in with your Eikon user ID and password.

`DfAddMonths`
`DfAddPeriod`
`DfAddWD`
`DfAddYears`
`DfAdjustToWD`
`DfCountDays`
`DfCountNonWD`
`DfCountWD`
`DfCountYears`
`DfFindDateD`
`DfFindDateM`
`DfFormatDate`
`DfIDNDate`
`DfIsWD`
`DfLastWD`
`DfListHolidays`

ErrorCode Property

The `ErrorCode` property of `AdxDateModule` stores the last error code generated by the object.

Syntax

```
ErrorCode() As Long
```

VBA sample

Example	Explanation
`Dim mo as AdxDateModule`	Declares an object of the type AdxDateModule
`Set mo = New AdxDateModule`	Creates an AdxDateModule object
`If mo.ErrorCode <> 0 Then`	mo.ErrorCode contains an error
`MsgBox (mo.ErrorCode)`	Displays the error code in a message box
`End If`	

Arguments

None

ErrorMode Property

The `ErrorMode` property of `AdxDateModule` sets the error mode of the `AdxModule`.

Modes

Keyword	In this mode
`DIALOGBOX`	a dialog box is displayed when an error occurs
`EXCEPTION`	an exception is thrown when an error occurs
`NO_EXCEPTION`	the user must call the property `ErrorCode` to know the error encountered

Syntax

```
ErrorMode() As AdxErrorMode
```

VBA sample

Example	Explanation
`Dim mo as AdxDateModule`	Declares an object of the type AdxDateModule
`Set mo = New AdxDateModule`	Creates an AdxDateModule object
`mo.ErrorMode = NO_EXCEPTION`	mo.ErrorMode is set to `NO_EXCEPTION`

Arguments

None

Default value

By default the error mode is set to `EXCEPTION`.

ErrorString Property

The `ErrorString` property of `AdxDateModule` stores the last error code generated by the object.

Syntax

```
ErrorString() As String
```

VBA sample

Example	Explanation
`Dim mo as AdxDateModule`	Declares an object of the type AdxDateModule
`Set mo = New AdxDateModule`	Creates an AdxDateModule object
`mo.ErrorMode = EXCEPTION`	mo.ErrorMode must be set to `EXCEPTION` to use the ErrorString
`On Error Go To Handle`	Error handler executes code after Handle when an error occurs
`RES = mo.DfIsWD("EUR", "05-MAR-01", "LAY:V")`	The calendar code is incorrect. The function DfIsWD returns an error
`Exit Sub`	Exits sub or function before the handle or code is executed
`Handle`	
`MsgBox(mo.ErrorString)`	Displays the ErrorString in a message box. In this case, the error string is 'Error#f000 – Calendars:Calendar(s) not found.'

Arguments

None

AdxEquityModule

You can use the `AdxEquityModule` functions to work with equities. These functions measure the relationship between risk and return and calculate the theoretical stock price as well as the internal rate of return or risk premium.

AdfinX Functions in AdxEquityModule

To find out more about these functions, consult the Adfin Library online help. You may be prompted to sign in with your Eikon user ID and password.

`AdEqDividendDiscountModel` `AdStatRegression`

ErrorCode Property

The `ErrorCode` property of `AdxEquityModule` stores the last error code generated by the object.

ErrorMode Property

The `ErrorMode` property of `AdxEquityModule` sets the error mode.

ErrorString Property

The `ErrorString` property of `AdxEquityModule` stores the last error code generated by the object.

AdxExoticModule

You can use `AdxExoticModule` functions to price exotic options. Some of these exotic options include Asian options, lookback options, and Bermudan options.

AdfinX Functions in AdxExoticModule

To find out more about these functions, consult the Adfin Library online help. You may be prompted to sign in with your Eikon user ID and password.

`AdOpDoubleNoTouchDeriv`	`AdOpDoubleNoTouchImpliedVol`	`AdOpDoubleNoTouchPremium`
`OpAsianDeriv`	`OpAsianImpliedVol`	`OpAsianPremium`
`OpBarrierDeriv`	`OpBarrierImpliedVol`	`OpBarrierPremium`
`OpBasketDeriv`	`OpBasketPremium`	`OpBinaryDeriv`
`OpBinaryImpliedVol`	`OpBinaryPremium`	`OpChooserDeriv`
`OpChooserImpliedVol`	`OpChooserPremium`	`OpCliquetDeriv`
`OpCliquetImpliedVol`	`OpCliquetPremium`	`OpCompoundDeriv`
`OpCompoundImpliedVol`	`OpCompoundPremium`	`OpDoubleBarrierDeriv`
`OpDoubleBarrierImpliedVol`	`OpDoubleBarrierPremium`	`OpExLookbackDeriv`
`OpExLookbackImpliedVol`	`OpExLookbackPremium`	`OpFxLinkedDeriv`
`OpFxLinkedImpliedVol`	`OpFxLinkedPremium`	`OpPowerDeriv`
`OpPowerImpliedVol`	`OpPowerPremium`	`OpRainbowDeriv`
`OpRainbowPremium`		

ErrorCode Property

The `ErrorCode` property of `AdxExoticModule` stores the last error code generated by the object.

Syntax

```
ErrorCode() As Long
```

VBA sample

Example	Explanation
`Dim ex as AdxExoticModule`	Declares an object of the type AdxExoticModule
`Set ex = New AdxExoticModule`	Creates an AdxExoticModule object
`If ex.ErrorCode <> 0 Then`	ex.ErrorCode contains an error
`MsgBox (ex.ErrorCode)`	Displays the error code in a message box
`End If`	

Arguments

None

ErrorMode Property

The `ErrorMode` property of `AdxExoticModule` sets the error mode.

Modes

Keyword	In this mode
`DIALOGBOX`	a dialog box is displayed when an error occurs
`EXCEPTION`	an exception is thrown when an error occurs
`NO_EXCEPTION`	the user must call the property ErrorCode to know the error encountered

Syntax

```
ErrorMode() As AdxErrorMode
```

VBA sample

Example	Explanation
`Dim ex as AdxExoticModule`	Declares an object of the type AdxExoticModule
`Set ex = New AdxExoticModule`	Creates an AdxExoticModule object
`ex.ErrorMode = NO_EXCEPTION`	ex.ErrorMode is set to `NO_EXCEPTION`

Arguments

None

Default value

By default the error mode is set to `EXCEPTION`.

ErrorString Property

The `ErrorString` property of `AdxExoticModule` retrieves the last error code generated by the object.

Syntax

```
ErrorString() As String
```

VBA sample

Example	Explanation
`Dim ex as AdxExoticModule`	Declares an object of the type AdxExoticModule
`Set ex = New AdxExoticModule`	Creates an AdxExoticModule object
`ex.ErrorMode = EXCEPTION`	ex.ErrorMode must be set to `EXCEPTION` to use the ErrorString
`On Error Go To Handle`	Error handler executes code after Handle when an error occurs
`Res = ex.OpBasketPremium("1-MAR-04", "31Aug04", [pricea].Value, "70", [corra].Value, "0.064", [returnar].Value, _[weighta].Value, "EXM:EURO", "RM:YTM", "CMTFORM"`	The CalcStructure is incorrect. The function OpBasketPremium returns an error
`Exit Sub` `Handle`	Exits sub or function before the handle or code is executed
`MsgBox(ex.ErrorString)`	Displays the ErrorString in a message box. In this case, the error string contains 'Error#7601 – CalcStructure: invalid keyword.'

Arguments

None

AdxForexModule

You can use `AdxForexModule` to handle calculation on the Forex and Money Markets.
These functions calculate and evaluate:

- cross rates
- swap points and outrights
- forward-forwards
- deposits
- FRAs

AdfinX Functions in AdxForexModule

To find out more about these functions, consult the Adfin Library online help. You may be prompted to sign in with your Eikon user ID and password.

`AdFxDepToSwp`
`AdFxSwpToDep`
`AdFxSwpToSwp`
`FxCalcPeriod`
`FxCross`
`FxCrossA`
`FxCrossD`
`FxGenCalc`
`FxGenParse`
`FxSwpToOut`

ErrorCode Property

The `ErrorCode` property of `AdxForexModule` stores the last error code generated by the object.

Syntax

```
ErrorCode() As Long
```

VBA sample

Example	Explanation
`Dim fx as AdxForexModule`	Declares an object of the type AdxForexModule
`Set fx = New AdxForexModule`	Creates an AdxForexModule object
`If fx.ErrorCode <> 0 Then`	fx.ErrorCode contains an error
`MsgBox (fx.ErrorCode)`	Displays the error code in a message box
`End If`	

Arguments

None

ErrorMode Property

The `ErrorMode` property of `AdxForexModule` sets the error mode of the `AdxForexModule`.

Modes

Keyword	In this mode
`DIALOGBOX`	a dialog box is displayed when an error occurs
`EXCEPTION`	an exception is thrown when an error occurs
`NO_EXCEPTION`	the user must call the property ErrorCode to know the error encountered

Syntax

```
ErrorMode() As AdxErrorMode
```

VBA sample

Example	Explanation
`Dim fx as AdxForexModule`	Declares an object of the type AdxForexModule
`Set fx = New AdxForexModule`	Creates an AdxForexModule object
`fx.ErrorMode = NO_EXCEPTION`	fx.ErrorMode is set to `NO_EXCEPTION`

Arguments

None

Default value

By default the error mode is set to `EXCEPTION`.

ErrorString

The `ErrorString` property of `AdxForexModule` retrieves the last error code generated by the object.

Syntax

```
ErrorString() As String
```

VBA sample

Example	Explanation
`Dim fx as AdxForexModule`	Declares an object of type AdxForexModule
`Set fx = New AdxForexModule`	Creates an AdxForexModule object
`fx.ErrorMode = EXCEPTION`	fx.ErrorMode must be set to `EXCEPTION` to use the ErrorString
`On Error Go To Handle`	Error handler executes code after Handle when an error occurs
`RES = fx.FxCurInfo("GFP", "LAY:V")`	
	The currency code `GFP` is incorrect. The function FxCurInfo returns an error
`Exit Sub`	Exits sub or function before the handle or code will be executed
`Handle`	
`MsgBox(fx.ErrorString)`	Displays the ErrorString in a message box
	fx.ErrorString contains the ErrorString ('Error#8200 – Currency: invalid currency code.')

Arguments

None

AdxModule

You can use `AdxModule` functions to work with the settings in the AdfinX functions library.

AdfinX Functions in AdxModule

To find out more about these functions, consult the Adfin Library online help. You may be prompted to sign in with your Eikon user ID and password.

`AdDefAttribute`	`AdDefSet`	`AdDefStructure`
`AdReadParam`	`AdStyleAttribute`	`AdStyleDelete`
`AdStyleName`	`AdStyleSet`	`AdStyleStructure`
`AdWriteParam`	`DfFormatDate`	`DfIDNDate`

DisplayStyles Method

The `DisplayStyles` method of `AdxModule` displays the style dialog box.

Syntax

```
DisplayStyles()
```

VBA sample

Example	Explanation
`Dim dlg as AdxModule`	Declares an object variable of the type AdxModule
`Set dlg = New AdxModule`	Creates an AdxModule object
`dlg.DisplayStyles`	Displays the styles dialog box

Arguments

None

SetReferenceDates Method

The `SetReferenceDates` method of `AdxModule` sets the offset (base day), the minimum date, and the maximum date.

Default values

The default values are 0, 1, 65380, since AdfinX valid dates must be between 31 December 1899 (1) and 31 December 2078 (65380).

Minimum date limit

The date minimum limit is 1 January 100 and the maximum limit is 31 December 3999.

Using the offset

The offset is used to increase the lower range for the dates that can be handled. If an error occurs while calling the function then the reference dates will be reset to the default values.

Syntax

The function can be used with `MaxDate` and `MinDate` as shown in the coding example to obtain the maximum and minimum dates set.

```
SetReferenceDates(DateOffset As Long, MinDate As Long, MaxDate As Long)
```

VBA sample

Example	Explanation
`Dim dlg as AdxModule`	Declares an object variable of the type AdxModule
`Set dlg = New AdxModule`	Creates an AdxModule object
`dlg.SetReferenceDates 0, 1, 65380`	Sets the reference dates
`MsgBox dlg.MinDate`	Gets the minimum date set
`MsgBox dlg.MaxDate`	Gets the maximum date

Arguments

Argument	Explanation
`DateOffset`	New offset (base day)
`MinDate`	New minimum date
`MaxDate`	New maximum date

ErrorCode Property

The `ErrorCode` property of `AdxModule` stores the last error code generated by the object.

Syntax

```
ErrorCode() As Long
```

VBA sample

Example	Explanation
`Dim mo as AdxModule`	Declares an object of the type AdxModule
`Set mo = New AdxModule`	Creates an AdxModule object
`If mo.ErrorCode <> 0 Then`	mo.ErrorCode contains an error
`MsgBox (mo.ErrorCode)`	Displays the error code in a message box
`End If`	

Arguments

None

ErrorMode Property

The `ErrorMode` property of `AdxModule` sets the error mode of the AdxModule.

Modes

Keyword	In this mode
`DIALOGBOX`	a dialog box is displayed when an error occurs
`EXCEPTION`	an exception is thrown when an error occurs
`NO_EXCEPTION`	the user must call the property ErrorCode to know the error encountered

Syntax

```
ErrorMode() As AdxErrorMode
```

VBA sample

Example	Explanation
`Dim mo as AdxModule`	Declares an object of the type AdxModule
`Set mo = New AdxModule`	Creates an AdxModule object
`mo.ErrorMode = NO_EXCEPTION`	mo.ErrorMode is set to `NO_EXCEPTION`

Arguments

None

Default value

By default the error mode is set to `EXCEPTION`.

ErrorString Property

The `ErrorString` property of `AdxModule` retrieves the last error code generated by the object.

Syntax

```
ErrorString() As String
```

VBA sample

Example	Explanation
`Dim mo as AdxModule`	Declares an object of the type AdxModule
`Set mo = New AdxModule`	Creates an AdxModule object
`mo.ErrorMode = EXCEPTION`	mo.ErrorMode must be set to `EXCEPTION` to use the ErrorString
`On Error Go To Handle`	Error handler executes code after Handle when an error occurs
`RES = mo.DfIsWD("EUR", "05-MAR-01", "LAY:V")`	The calendar code is incorrect. The function DfIsWD returns an error
`Exit Sub` `Handle`	Exits sub or function before the handle or code is executed
`MsgBox(mo.ErrorString)`	Displays the ErrorString in a message box. In this case, the error string is 'Error#f000 – Calendars:Calendar(s) not found.'

Arguments

None

MaxDate Property

The `MaxDate` method of `AdxModule` returns the maximum date value set.

Syntax

```
MaxDate() As Long
```

VBA sample

Example	Explanation
`Dim dlg as AdxModule`	Declares an object variable of the type AdxModule
`Set dlg = New AdxModule`	Creates an AdxModule object
`MsgBox dlg.MaxDate`	Gets the maximum date

Arguments

None

Return value

A `Long` value, containing the maximum date.

MinDate Property

The `MinDate` method of `AdxModule` returns the minimum date value set.

Syntax

```
MinDate() As Long
```

VBA sample

Example	Explanation
`Dim dlg as AdxModule`	Declares an object variable of the type AdxModule
`Set dlg = New AdxModule`	Creates an AdxModule object
`MsgBox dlg.MinDate`	Gets the minimum date

Arguments

None

Return value

A `Long` value, containing the minimum date.

AdxOptionModule

You can use `AdxOptionModule` functions to calculate premiums volatilities, and common derivatives (commonly known as the Greeks). These functions calculate the:

- premium
- historical and implied volatility
- derivatives (delta, gamma, vega, theta, rho)
- gearing
- break-even time

AdfinX Functions in AdxOptionModule

To find out more about these functions, consult the Adfin Library online help. You may be prompted to sign in with your Eikon user ID and password.

AdBarrierCapFloorBSCaplets	AdBarrierCapFloorBSDeriv	AdBarrierCapFloorBSImpliedStrike
AdBarrierCapFloorBSPremium	AdBarrierCapFloorCaplets	AdBarrierCapFloorDeriv
AdBarrierCapFloorImpliedVol	AdBarrierCapFloorPremium	AdBondOptionDeriv
AdBondOptionPremium	AdCapFloorBSCaplets	AdCapFloorBSDeriv
AdCapFloorBSImpliedStrike	AdCapFloorBSPremium	AdCapFloorBSVol
AdCapFloorCaplets	AdCapFloorDeriv	AdCapFloorImpliedVol
AdCapFloorPremium	AdCapFloorVolSurface	AdCdsOptionBSDeriv
AdCdsOptionBSPremium	AdCmsCapFloorDeriv	AdCmsCapFloorImpliedVol
AdCmsCapFloorPremium	AdCmsSpreadOptionCaplets	AdCmsSpreadOptionDeriv
AdCmsSpreadOptionImpVol	AdCmsSpreadOptionPremium	AdDigitalCapFloorBSCaplets
AdDigitalCapFloorBSDeriv	AdDigitalCapFloorBSImpliedStrike	AdDigitalCapFloorBSPremium
AdDigitalCapFloorCaplets	AdDigitalCapFloorDeriv	AdDigitalCapFloorImpliedVol
AdDigitalCapFloorPremium	AdFxVolSurface	AdOpDoubleNoTouchDeriv
AdOpDoubleNoTouchImpliedVol	AdOpDoubleNoTouchPremium	AdOpVannaVolgaAdjustment
AdSwaptionBSDeriv	AdSwaptionBSImpliedVol	AdSwaptionBSPremium
AdSwaptionDeriv	AdSwaptionPremium	OpCalcDeriv
OpHistVol	OpImpliedStrike	OpImpliedVol
OpPremium		

ErrorCode Property

The `ErrorCode` property of `AdxScheduleModule` stores the last error code generated by the object.

Syntax

```
ErrorCode() As Long
```

VBA sample

Example	Explanation
`Dim op as AdxScheduleModule`	Declares an object of the type AdxScheduleModule
`Set op = New AdxScheduleModule`	Creates an AdxScheduleModule object
`If op.ErrorCode <> 0 Then`	op.ErrorCode contains an error
`MsgBox (op.ErrorCode)`	Displays the error code in a message box
`End If`	

Arguments

None

ErrorMode Property

The `ErrorMode` property of `AdxScheduleModule` sets the error mode of the `AdxScheduleModule`.

Modes

Keyword	In this mode
`DIALOGBOX`	a dialog box is displayed when an error occurs
`EXCEPTION`	an exception is thrown when an error occurs
`NO_EXCEPTION`	the user must call the property ErrorCode to know the error encountered

Syntax

```
ErrorMode() As AdxErrorMode
```

VBA sample

Example	Explanation
`Dim op as AdxScheduleModule`	Declares an object of the type AdxScheduleModule
`Set op = New AdxScheduleModule`	Creates an AdxScheduleModule object
`op.ErrorMode = NO_EXCEPTION`	op.ErrorMode is set to `NO_EXCEPTION`

Arguments

None

Default value

By default the error mode is set to `EXCEPTION`.

ErrorString Property

The `ErrorString` property of `AdxScheduleModule` retrieves the last error code generated by the object.

Syntax

```
ErrorString() As String
```

VBA sample

Example	Explanation
`Dim op as AdxScheduleModule`	Declares an object of the type AdxScheduleModule
`Set op = New AdxScheduleModule`	Creates an AdxScheduleModule object
`op.ErrorMode = EXCEPTION`	op.ErrorMode must be set to `EXCEPTION` to use the ErrorString
`On Error Go To Handle`	Error handler executes code after Handle when an error occurs
`RES = op.OpPremium("24-APR-97", "25-FEB-96", "1.69", "1.55", "1.55", "0.1976", "0.0197", "0.0207", "CALL EXM:AMER")`	The maturity date is before the calculation date. The function OpPremium returns an error

Example	Explanation
`Exit Sub` `Handle`	Exits sub or function before the handle or code is executed
`MsgBox(op.ErrorString)`	Displays the ErrorString in a message box. In this case, the error string is 'Error#7012 – Invalid maturity date.'

Arguments

None

AdxScheduleModule

The `AdxScheduleModule` class contains only properties related to error handling.

ErrorCode Property

The `ErrorCode` property of `AdxScheduleModule` stores the last error code generated by the object.

Syntax

```
ErrorCode() As Long
```

VBA sample

Example	Explanation
`Dim op as AdxScheduleModule`	Declares an object of the type AdxScheduleModule
`Set op = New AdxScheduleModule`	Creates an AdxScheduleModule object
`If op.ErrorCode <> 0 Then`	op.ErrorCode contains an error
`MsgBox (op.ErrorCode)`	Displays the error code in a message box
`End If`	

Arguments

None

ErrorMode Property

The `ErrorMode` property of `AdxScheduleModule` sets the error mode of the `AdxScheduleModule`.

Modes

Keyword	In this mode
`DIALOGBOX`	a dialog box is displayed when an error occurs
`EXCEPTION`	an exception is thrown when an error occurs
`NO_EXCEPTION`	the user must call the property ErrorCode to know the error encountered

Syntax

```
ErrorMode() As AdxErrorMode
```

VBA sample

Example	Explanation
`Dim op as AdxScheduleModule`	Declares an object of the type AdxScheduleModule
`Set op = New AdxScheduleModule`	Creates an AdxScheduleModule object
`op.ErrorMode = NO_EXCEPTION`	op.ErrorMode is set to `NO_EXCEPTION`

Arguments

None

Default value

By default the error mode is set to `EXCEPTION`.

ErrorString Property

The `ErrorString` property of `AdxScheduleModule` retrieves the last error code generated by the object.

Syntax

```
ErrorString() As String
```

VBA sample

Example	Explanation
`Dim op as AdxScheduleModule`	Declares an object of the type AdxScheduleModule
`Set op = New AdxScheduleModule`	Creates an AdxScheduleModule object
`op.ErrorMode = EXCEPTION`	op.ErrorMode must be set to `EXCEPTION` to use the ErrorString

Example	Explanation
`On Error Go To Handle`	Error handler executes code after Handle when an error occurs
`RES = op.OpPremium("24-APR-97", "25-FEB-96", "1.69", "1.55", "1.55", "0.1976", "0.0197", "0.0207", "CALL EXM:AMER")`	The maturity date is before the calculation date. The function OpPremium returns an error
`Exit Sub` `Handle`	Exits sub or function before the handle or code is executed
`MsgBox(op.ErrorString)`	Displays the ErrorString in a message box. In this case, the error string is 'Error#7012 – Invalid maturity date.'

Arguments

None

AdxSwapModule

You can use the `AdxswapModule` functions for:

- net present value calculation
- cash flow generation
- interest rate swap derivatives calculation
- swap solving
- sensitivity calculations

AdfinX Functions in AdxSwapModule

ⓘ To find out more about these functions, consult the Adfin Library online help. You may be prompted to sign in with your Eikon user ID and password.

`AdAssetSwapBdCashFlows`	`AdAssetSwapBdPrice`	`AdAssetSwapBdSpread`
`AdAssetSwapGenCashFlows`	`AdAssetSwapGenPrice`	`AdAssetSwapGenSpread`
`AdCancellableSwapDeriv`	`AdCancellableSwapNpv`	`AdCancellableSwapSolve`
`AdCBSToSwp`	`AdCmsCashFlows`	`AdCmsDeriv`
`AdCmsNpv`	`AdCmsSolve`	`AdInflationAssetSwapCashFlows`
`AdInflationAssetSwapPrice`	`AdInflationAssetSwapSpread`	`AdInflationSwapCashFlows`
`AdInflationSwapDeriv`	`AdInflationSwapNpv`	`AdInflationSwapSolve`
`AdIrsDeriv`	`AdQuantoSwapCashFlows`	`AdQuantoSwapDeriv`
`AdQuantoSwapNpv`	`AdQuantoSwapSolve`	`AdRangeAccrualSwapCashFlows`
`AdRangeAccrualSwapDeriv`	`AdRangeAccrualSwapNpv`	`AdRangeAccrualSwapSolve`
`SwCsCashFlows`	`SwCsPx`	`SwCsSolve`
`SwIrsCashFlows`	`SwIrsCpnDates`	`SwIrsPx`
`SwIrsSolve`	`SwSwpExtend`	

ErrorCode Property

The `ErrorCode` property of `AdxSwapModule` stores the last error code generated by the object.

Syntax

```
ErrorCode() As Long
```

VBA sample

Example	Explanation
`Dim sw as AdxSwapModule`	Declares an object of the type AdxSwapModule
`Set sw = New AdxSwapModule`	Creates an AdxSwapModule object
`If sw.ErrorCode <> 0 Then`	sw.ErrorCode contains an error
`MsgBox (sw.ErrorCode)`	Displays the error code in a message box
`End If`	

Arguments

None

ErrorMode Property

The `ErrorMode` property of `AdxSwapModule` sets the error mode of the `AdxSwapModule`.

Modes

Keyword	In this mode
`DIALOGBOX`	a dialog box is displayed when an error occurs
`EXCEPTION`	an exception is thrown when an error occurs
`NO_EXCEPTION`	the user must call the property ErrorCode to know the error encountered

Syntax

```
ErrorMode() As AdxErrorMode
```

VBA sample

Example	Explanation
`Dim sw as AdxSwapModule`	Declares an object of the type AdxSwapModule
`Set sw = New AdxSwapModule`	Creates an AdxSwapModule object
`sw.ErrorMode = NO_EXCEPTION`	sw.ErrorMode is set to `NO_EXCEPTION`

Arguments

None

Default value

By default the error mode is set to `EXCEPTION`.

ErrorString Property

The `ErrorString` property of `AdxSwapModule` stores the last error code generated by the object.

Syntax

```
ErrorString() As String
```

VBA sample

Example	Explanation
`Dim sw as AdxSwapModule`	Declares an object of the type AdxSwapModule
`Set sw = New AdxSwapModule`	Creates an AdxSwapModule object
`sw.ErrorMode = EXCEPTION`	sw.ErrorMode must be set to `EXCEPTION` to use the ErrorString
`On Error Go To Handle`	Error handler executes code after Handle when an error occurs.
`Res = sw.SwIrsCpnDates("1-MAR-04", "31Aug04", "1-Jun-02", "LBOTH FRQ:1 ACC:AA", "LAY:V")`	The `RES` keyword is not in `AdMode (1)`. The function SwIrsCpnDates returns an error.
`Exit Sub`	Exits sub or function before the handle or code is executed.
`Handle` `MsgBox(sw.ErrorString)`	Displays the ErrorString in a message box. In this case the error string is 'Error#40cb – SwMode: invalid value for RES keyword.'

Arguments

None

AdxUtilityModule

You can use `AdxUtilityModule` to handle calculations on the Forex and Money Markets, as well as to format and parse data.

AdfinX Functions in AdxUtilityModule

To find out more about these functions, consult the Adfin Library online help. You may be prompted to sign in with your Eikon user ID and password.

`AdDepToFra`
`AdDepToFraBA`
`AdFormat`
`AdHistoryUpdate`
`AdHistoryValue`
`AdInterp`
`AdInterpolation`
`AdParse`
`AdRateConv`
`AdRound`
`AdZcToFraBA`
`NormalC`
`NormalS`

ErrorCode Property

The `ErrorCode` property of `AdxUtilityModule` stores the last error code generated by the object.

Syntax

```
ErrorCode() As Long
```

VBA sample

Example	Explanation
`Dim mo as AdxUtilityModule`	Declares an object of the type AdxUtilityModule
`Set mo = New AdxUtilityModule`	Creates an AdxUtilityModule object
`If mo.ErrorCode <> 0 Then`	mo.ErrorCode contains an error
`MsgBox (mo.ErrorCode)`	Displays the error code in a message box
`End If`	

Arguments

None

ErrorMode Property

The `ErrorMode` property of `AdxUtilityModule` sets the error mode of the `AdxModule`.

Modes

Keyword	In this mode
`DIALOGBOX`	a dialog box is displayed when an error occurs
`EXCEPTION`	an exception is thrown when an error occurs
`NO_EXCEPTION`	the user must call the property ErrorCode to know the error encountered

Syntax

```
ErrorMode() As AdxErrorMode
```

VBA sample

Example	Explanation
`Dim mo as AdxUtilityModule`	Declares an object of type AdxUtilityModule
`Set mo = New AdxUtilityModule`	Creates an AdxUtilityModule object
`mo.ErrorMode = NO_EXCEPTION`	mo.ErrorMode is set to `NO_EXCEPTION`

Arguments

None

Default value

By default the error mode is set to `EXCEPTION`.

ErrorString Property

The `ErrorString` property of the `AdxUtilityModule` stores the last error code generated by the object.

Syntax

```
ErrorString() As String
```

VBA sample

Example	Explanation
`Dim mo as AdxModule`	Declares an object of type AdxModule
`Set mo = New AdxModule`	Creates an AdxModule object
`mo.ErrorMode = EXCEPTION`	mo.ErrorMode must be set to `EXCEPTION` to use the ErrorString
`On Error Go To Handle`	Error handler executes code after Handle when an error occurs
`RES = mo.DfIsWD("EUR", "05-MAR-01", "LAY:V")`	The calendar code is incorrect. The function DfIsWD returns an error
`Exit Sub` `Handle`	Exits sub or function before the handle or code is executed
`MsgBox(mo.ErrorString)`	Displays the ErrorString in a message box. In this case, the error string is *('Error#f000 – Calendars:Calendar(s) not found.')*

Arguments

None

AdxYieldCurveModule

You can use `AdxYieldCurveModule` to calculate periods and yield curves.

AdfinX Functions in AdxYieldCurveModule

To find out more about these functions, consult the Adfin Library online help. You may be prompted to sign in with your Eikon user ID and password.

`AdCalibrate`	`AdceForwardTermStructure`	`AdFutCodes`
`AdFutDates`	`AdGenTermStructure`	`AdInflationTermStructure`
`AdRate`	`AdTermStructure`	

ErrorCode Property

The `ErrorCode` property of `AdxYieldCurveModule` stores the last error code generated by the object.

Syntax

```
ErrorCode() As Long
```

VBA sample

Example	Explanation
`Dim yc as AdxYieldCurveModule`	Declares an object of the type AdxYieldCurveModule
`Set yc = New AdxYieldCurveModule`	Creates an AdxYieldCurveModule object
`If yc.ErrorCode <> 0 Then`	yc.ErrorCode contains an error
`MsgBox (yc.ErrorCode)`	Displays the error code in a message box
`End If`	

Arguments

None

ErrorMode Property

The `ErrorMode` property of `AdxYieldCurveModule` sets the error mode.

Modes

Keyword	In this mode
DIALOGBOX	a dialog box is displayed when an error occurs.
EXCEPTION	an exception is thrown when an error occurs.
NO_EXCEPTION	the user must call the property ErrorCode to know the error encountered.

Syntax

```
ErrorMode() As AdxErrorMode
```

VBA sample

Example	Explanation
Dim yc as AdxYieldCurveModule	Declares an object of the type AdxYieldCurveModule
Set yc = New AdxYieldCurveModule	Creates an AdxYieldCurveModule object
yc.ErrorMode = NO_EXCEPTION	yc.ErrorMode is set to NO_EXCEPTION

Arguments

None

Default value

By default the error mode is set to EXCEPTION.

ErrorString Property

The ErrorString property of AdxYieldCurveModule retrieves the last error code generated by the object.

Syntax

```
ErrorString() As String
```

VBA sample

Example	Explanation
`Dim yc as AdxYieldCurveModule`	Declares an object of the type AdxYieldCurveModule.
`Set yc = New AdxYieldCurveModule`	Creates an AdxYieldCurveModule object
`yc.ErrorMode = EXCEPTION`	yc.ErrorMode must be set to `EXCEPTION` to use the ErrorString.
`On Error Go To Handle`	Error handler executes code after Handle when an error occurs
`Res = yc.AdFutDates("CLDR:CAN YB:360 CUR:CAB NBQC:8 NBMC:3", "Mo", "LAY:V")`	The maturity code should be `MO`. The function `AdFutDates` in AdfinX returns an error.
`Exit Sub` `Handle`	Exits sub or function before the handle or code is executed.
`MsgBox(yc.ErrorString)`	Displays the ErrorString in a message box. In this case, the error string is 'Error#4009 – Maturity code: invalid value.'

Arguments

None

AdXErrorMode

Keyword	In this mode
DIALOGBOX	a dialog box is displayed when an error occurs.
EXCEPTION	an exception is thrown when an error occurs.
NO_EXCEPTION	the user must call the property ErrorCode to see the error encountered.

Syntax

```
AdxErrorMode
```

AdfinX 6.0 Real-Time Library

AdfinX Real-Time Library Overview

Real-Time Data Access

Overview

The AdfinX Real-Time Component Library (AdxRtxLib) provides a number of components allowing user applications to retrieve and contribute real-time data sources. The components that AdxRtxLib supplies provide dual interfaces.

Data object components

Four data object components are provided. These are `AdxRtList`, `AdxRtContribute`, `AdxRtHistory` and `AdxRtChain`.

Access Settings

The settings file

The file `overrideconfiguration.xml` holds AdfinX Real Time and other user data settings.

Configuration

To configure these settings, refer to the full documentation under *Configuration Appendix > Configuring General Settings > Introduction to Configuring General Settings* in Thomson Reuters Eikon Deployment Help.

Referencing the ActiveX Controls

Multi-platform access

You can access the ActiveX controls from various programming platforms. This section provides examples of how they can be set up.

How to set up VBA in Thomson Reuters Eikon – Microsoft Office

1 Open Microsoft Excel (or Word or PowerPoint).

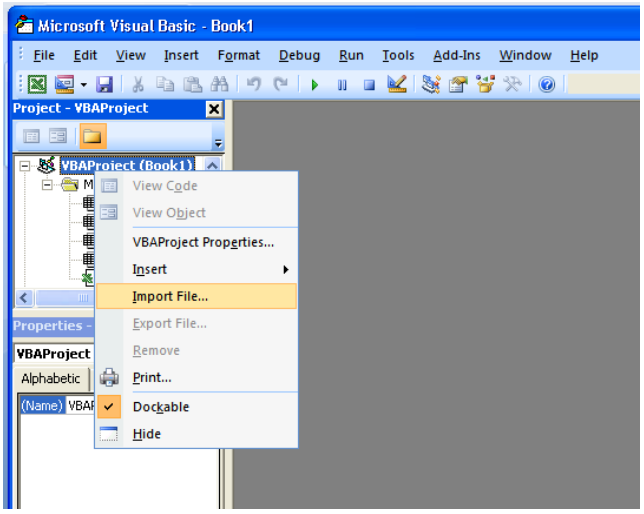
2 Click *Developer > Visual Basic* OR press *Alt+F11*

The VBA screen opens.

The Developer tab is hidden in Microsoft Excel 2007. To display this tab, see "How to display the Developer tab" on page 149.

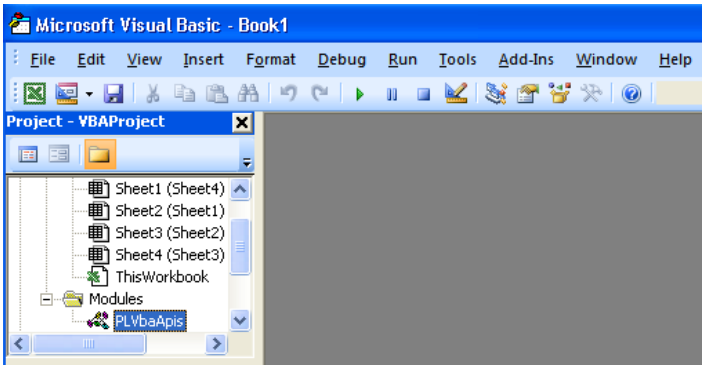
3 Right-click *VBAProject (Bookn)* and choose *Import File*.

The *Import File* window opens.



4 Select *PLVbaApis.bas* from the Thomson Reuters Eikon installation folder and click *Open*.

The module is added to the VBA project.



5 In the VBA project, click *Tools > References* to add the references to Thomson Reuters Eikon APIs.
   - RTX.DLL
   - ADXOO.DLL
   - ADXFO.DLL
   - RSEARCH.DLL
   - DEX2.DLL

Note
If they do not appear by default, browse to `C:\Program Files (x86)\Thomson Reuters\Eikon\X\Bin\`

And use the following folder for DEX2: `C:\Program Files (x86)\Thomson Reuters\Eikon\X\Bin\Apps\TR.OFFICE.CORE\0.0.0.0\Bin\`

All platforms other than Thomson Reuters Eikon – Microsoft Office

For all platforms other than Thomson Reuters Eikon – Microsoft Office, see the *EikonDesktopDataAPI* documentation on the *TR Professional Developer Community portal*.

AdxRtxLib Overview

The AdxRtxLib component provides a way of accessing real-time market data from within Windows applications using standard ActiveX technology.

Real-time data items

Real-time market data are organized as items, which are in turn composed of fields that hold individual atomic values. An item might represent the information concerning an individual bond, for example.

Items are themselves provided by data sources. When the data in an item needs to change, its source updates the required fields, keeping their values up to date.

Real-time market data

The AdxRtxLib component allows a program to retrieve field values from market data items and can inform the program when the source changes them. Using this component, the program can perform a variety of tasks:

- Maintaining lists of items providing real-time market data
- Managing the list of fields within those items whose values are required
- Handling updates made to field values by the data source
- Allowing the retrieval of a single item image ("snapshot"), which will not be kept up to date
- Allowing the retrieval of historical market data for some data sources

Contribution

Some data sources also allow contribution, whereby new values are supplied for item fields. All consumers of the source will be informed of updates to those items in the usual way. Once again, the AdxRtxLib component allows a program to contribute data in this way.

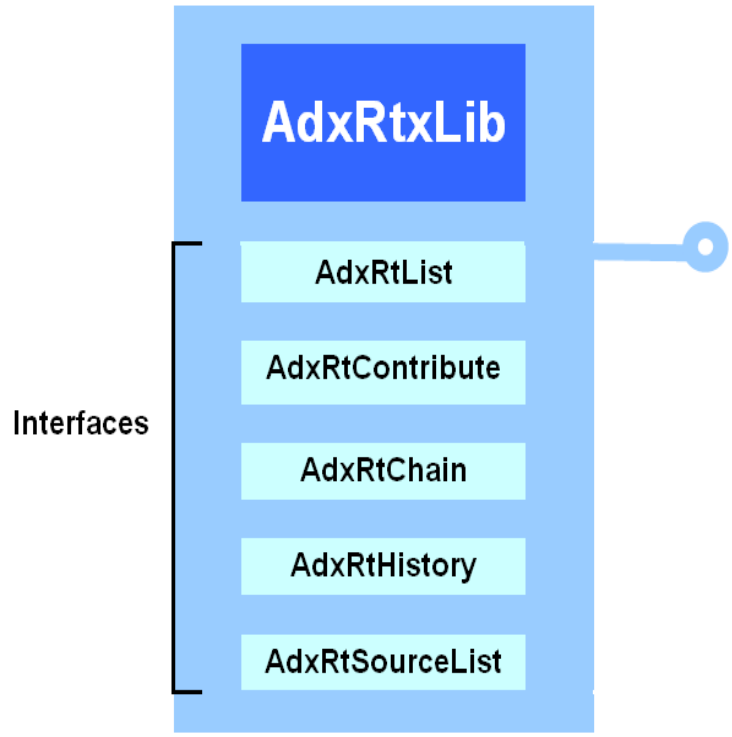
The AdxRtxLib Objects

To support this functionality within a program, the AdxRtxLib component provides objects of the following types:

Objects	Description
AdxRtList	Represents a collection of items, every item being a collection of fields. By iterating through the collection, the program can retrieve information about any item or any field.
AdxRtContribute	Allows the program to contribute new values for a set of fields in an item.
AdxRtHistory	Allows the program to request historical data concerning an item.
AdxRtChain	Allows the program to retrieve chains, which are lists of item names provided by the data source.
AdxRtSourceList	Allows the program to retrieve information about sources and information about all the fields that are available.

AdxRtxLib architecture

This figure illustrates the AdxRtxLib architecture:



Library DLL file

The AdxRtxLib component functionality is provided using ActiveX interfaces. These are implemented by the `RTX.DLL` file.

AdxRtList

AdxRtList Properties

AdxRtList Attribute

Allows the consultation of various settings in the object. These are the settings that can be changed for the object by assigning values to the AdxRtList Mode property.

Arguments

`AttrID` An identification of the attribute, passed as a Variant. This can be either an identifying code taken from the AdxRtList Mode enumeration or a string containing the keyword used to set the attribute using the AdxRtList Mode property.

Return value

The value of the requested attribute of type Variant.

AdxRtList DebugLevel

This property defines the support for debugging provided by the AdxRtList object. It takes a value from the RT_DebugLevel enumeration.

Arguments

None

AdxRtList ErrorCode

This property retrieves the error code of the latest error encountered by the object as an integer value. If the ErrorMode property of the object is set to NO_EXCEPTION (see "AdxErrorMode" on page 106). This value must be consulted by the client application to determine whether an error has occurred.

Arguments

None

AdxRtList ErrorMode

This property sets the error mode for the object. When an error is detected, this property is consulted to see what action should be taken by the object to inform the user of the application. It must be assigned a value from the AdxErrorMode enumeration.

Default value

By default the error mode is set to EXCEPTION. See "AdxErrorMode" on page 106.

Arguments

None

AdxRtList ErrorString

This property retrieves a string describing the latest error encountered by the object. If the `ErrorMode` property of the object is set to NO_EXCEPTION (see "AdxErrorMode" on page 106). This value can be consulted by the client application when the `ErrorCode` property indicates that an error has occurred.

Arguments

None

AdxRtList FieldListCount

This property supplies the number of fields described by the AdxRtList ListFields property for a particular item.

Arguments

`ItemName` The name of the required item as a String.

`RView`	One of the values from the enumeration. This specifies how the counting should be performed: one row for each updated field, one row for each registered field, or one row for each field supplied by the identified item.

Return value

The number of fields counted is supplied as an integer. This is zero if an error occurs.

Error handling

An error exception may be thrown in the following cases:

If `RView` is set to RT_FRV_UPDATED (see RT_FieldRowView) and the property is queried outside of an event callback (i.e. one of AdxRtList OnImage, AdxRtList OnUpdate or AdxRtList OnTime), the return value is 0. In this case, an error exception is thrown.

If `RView` contains an invalid value, an error exception is thrown.

VBA sample

```
Public list as AdxRtList
Set list = CreateAdxRtList
Dim MyArray as Variant
MyArray = list.ListFields("DEM=", RT_
FRV_ALL, RT_FCV_VALUE
NumFields = list.FieldListCount("DEM=",     'NumFields holds the number of rows (fields)
RT_FRV_ALL                                  contained in MyArray. '
```

See also

"AdxRtList ItemListCount" on page 60
"AdxRtList ListFields" on page 61
"AdxRtList ListItems" on page 63

AdxRtList FieldStatus

This property returns a value from the RT_FieldStatus enumeration.

Arguments

`ItemName`	The name of the item as a String.
`FieldName`	The name of the field as a String.

Return value

Returns the status of the field in the item.

Error handling

If the AdxRtList Source is not set (i.e. the source is in the state `RT_SOURCE_NOT_SET`, see "RT_SourceStatus" on page 103), an error exception is thrown.

VBA sample

```
Public list as AdxRtList
Set list = CreateAdxRtList
Dim ItemList(0 To 1) as Variant
ItemList(0) = "EUR="
ItemList(1) = "JPY="
Dim FieldList(0 To 1) as Variant
FieldList(0) = "BID"
FieldList(1) = "ASK"
list.RegisterItems ItemList,
FieldList
list.StartUpdates RT_MODE_IMAGE
Data = list.value("EUR=", "BID")
Fstatus = list.FieldStatus("EUR=",
"BID")                          'Sets FStatus to RT_FIELD_OK or RT_FIELD_
                                UNKNOWN'
```

AdxRtList ItemListCount

This property supplies the number of items described by the "AdxRtList ListItems" on page 63 property.

Arguments

`RView` One of the values from the "RT_ItemRowView" on page 103 enumeration. This specifies how the counting should be performed: one row for each updated field, one row for each registered field, or one row for each field supplied by the identified item.

Return value

The number of items counted is supplied as an integer. This is zero if an error occurs.

Error handling

An error exception may be thrown in the following cases:

If `RView` is set to RT_IRV_UPDATED (see "RT_ItemRowView" on page 103) and the property is queried outside of an event callback (i.e. one of AdxRtList OnImage, AdxRtList OnUpdate or AdxRtList OnTime), the return value is 0. In this case, an error exception is thrown.

If `RView` contains an invalid value, an error exception is thrown.

VBA sample

```
Public list as AdxRtList
Set list = CreateAdxRtList
Dim MyArray as Variant
```

```
MyArray = list.ListItems(RT_IRV_ALL,
RT_ICV_USERTAG
ListCount = list.ItemListCount(RT_IRV_ 'ListCount holds the number of rows (items)
ALL) contained in MyArray. '
```

See also

"AdxRtList FieldListCount" on page 58
"AdxRtList ListFields" on page 61
"AdxRtList ListItems" on page 63

AdxRtList ItemStatus

This property returns a value from the RT_ItemStatus enumeration.

Arguments

ItemName	The name of the item as a String.

Return value

Returns the status of the item.

Error handling

AdxRtList Source is not set (i.e. the source is in the state RT_SOURCE_NOT_SET, see "RT_SourceStatus" on page 103), an error exception is thrown.

VBA sample

```
Public list as AdxRtList
Set list = CreateAdxRtList
Dim ItemList(0 To 1) as
Variant
ItemList(0) = "EUR="
ItemList(1) = "JPY="
Dim FieldList(0 To 1) as
Variant
FieldList(0) = "BID
FieldList(1) = "ASK"
list.RegisterItems ItemList,
FieldList
IStatus = list.ItemStatus 'Sets IStatus to the status of the item, such as RT_ITEM_OK
("EUR=", "BID") or RT_ITEM_STALE'
```

AdxRtList ListFields

This property supplies the field information of an item as a two dimensional array of the type Variant. The array can have up to four columns. The first column identifies the fields. Other, optional columns supply field data

values, field status (see the RT_FieldStatus enumeration) and user tag (see the AdxRtList UserTag property) values.

The `RView` and `CView` parameters control the contents of the array.

Arguments

`RView`	One of the values from the RT_FieldRowView enumeration. This specifies that the array should contain a row for either: • each field supplied by the item, or • each registered field of the item, or • only each updated field of the item (changed since the last update was handled).
`ItemName`	The name of the required item as a `String`.
`CView`	A combination of values from the RT_FieldColumnView enumeration. This specifies which of the three optional columns will be supplied: data, status, and user tag. The enumeration values should be combined by adding them together.

Return value

The property is supplied as a Variant array of fields or an empty Variant if an error has occurred.

Error handling

This table describes the cases in which an error exception may be thrown:

If	Then
• `RView` is set to `RT_FRV_UPDATED`RT_FieldRowView • the property is queried outside of an event callback (i.e. one of AdxRtList OnImage, AdxRtList OnUpdate or AdxRtList OnTime)	An error exception is thrown because the return array is empty.
• `RView` is set to RT_FRV_EXISTING (see RT_FieldRowView • `CView` includes the RT_FCV_USERTAG flag (see RT_FieldColumnView)	An error exception is thrown because user tags can be set only for registered fields.
Either `RView` or `CView` contain invalid values	An error exception is thrown.

Notes

- By default, RView is set to RT_FRV_ALL (see RT_FieldRowView), and CView to the combination (RT_FCV_VALUE (see RT_FieldColumnView) + RT_FCV_USERTAG (see RT_FieldColumnView).
- RT_FRV_EXISTING mode (see RT_FieldRowView) is only valid with certain platforms such as SSL.

VBA sample

```
Public list as AdxRtList

Set list = CreateAdxRtList

Dim MyArray as Variant

MyArray = list.ListFields ("DEM=", RT_FRV_ALL, RT_FCV_VALUE
                                    'The returned array contains the names and values of all fields available in the "DEM=" item. '
```

`FieldName = MyArray(2, 0)`	'FieldName contains the value of the third element of the first column of the array; this is the name of the third field. '
`FieldData = MyArray(2, 1)`	'FieldData contains the value of the third element of the second column of the array; this is the value of the third field. '

See also

"AdxRtList FieldListCount" on page 58
"AdxRtList ItemListCount" on page 60
"AdxRtList ListItems" on page 63

AdxRtList ListItems

This property supplies item information in a two dimensional array of the type Variant. The array can have up to three columns:

- item identification
- item status (optional, see the RT_ItemStatus enumeration)
- user tag values for each item value (optional, see the AdxRtList UserTag property)

Arguments

Argument	Description	Default value
`RView`	One of the values from the RT_ItemRowView enumeration. This specifies that the array should contain a row for each registered item, or only a row for each item that is updated.	RT_IRV_ALL (see "RT_ItemRowView" on page 103)
`CView`	A combination of values from the RT_ItemColumnView enumeration. This specifies which of the two optional columns will be supplied: status and user tag. The enumeration values should be combined by adding them together.	RT_ICV_USERTAG (see "RT_ItemColumnView" on page 104)

Return value

The property is supplied as a Variant array of items or an empty Variant if an error has occurred.

Error handling

An error exception may be thrown in the following cases:

- If `RView` is set to RT_IRV_UPDATED (see "RT_ItemRowView" on page 103) and the property is queried outside of an event callback (i.e. one of AdxRtList OnImage, AdxRtList OnUpdate or AdxRtList OnTime), the return array will be empty. In this case, an error exception is thrown.
- If either `RView` or `CView` contain invalid values, an error exception is thrown.

Notes

The RT_IRV_ALL mode (see "RT_ItemRowView" on page 103)is only valid with certain platforms (all except DDE).

VBA sample

```
Public list as AdxRtList
```

```
Set list = CreateAdxRtList
Dim MyArray as Variant

MyArray = list.ListItems(RT_IRV_UPDATED, RT_ICV_USERTAG   'The returned array contains the names and user tags of all updated items in list. '
ItemName = MyArray(2, 0)                                  'ItemName contains the value of the third element of the first column of the array; this is the name of the third item.
ItemTag = MyArray(2, 1)                                   'ItemTag contains the value of the third element of the second column of the array; this is the user tag of the third item.
```

See also

"AdxRtList FieldListCount" on page 58

"AdxRtList ItemListCount" on page 60

"AdxRtList ListFields" on page 61

AdxRtList ListStatus

This property returns a value from the RT_ListStatus enumeration. This value reflects the status of the AdxRtList object itself.

Arguments

None

Return value

Returns the status of the list.

Error handling

If the AdxRtList Source is not set (i.e. the source is in the state RT_SOURCE_NOT_SET, see "RT_SourceStatus" on page 103), an error exception is thrown.

VBA sample

```
Public list as AdxRtList
Set list = CreateAdxRtList
Status = list.ListStatus      'Sets Status to the status of the list object; this is RT_LIST_INACTIVE if no items are registered. '
```

AdxRtList Mode

This property describes various behavioral aspects of the AdxRtList object. In particular, it determines how the list object delivers data update events to the client application. The parameters are:

Parameter	Description	Possible values	Default value
FRQ	The frequency of regularly generated events. If the AdxRtList StartUpdates method of the list was called with the run mode value RT_MODE_ONTIMEor RT_MODE_ONTIME_IF_UPDATEDRT_RunMode, AdxRtList OnTime events will be triggered at this frequency. The minimum frequency is one event per second.	FRQ:iH sets a frequency of one event every i hours. FRQ:iM sets a frequency of one event every i minutes. FRQ:iS sets a frequency of one event every i seconds.	One event every 10 seconds.
TIMEOUT	The maximum waiting time for the list data.	TIMEOUT:i sets a timeout period of i seconds.	Zero, indicating that no timeout is in effect.
AdjustToSystemTimeZone	The TIME field display option.	AdjustToSystemTimeZone:Yes converts the TIME value to local time. AdjustToSystemTimeZone:No uses the TIME value as provided by the market infrastructure, usually as GMT.	

Specifying more than one mode

To specify more than one mode at once, use a space to delimit the parameters. For example:

```
list.Mode = "FRQ:10s AdjustToSystemTimeZone:Yes"
```

Arguments

None

Mode

```
Public list as AdxRtList                    'Declare as global variable'
list.Source = "IDN"                         'Set the source name to "IDN"
Set list = CreateAdxRtList                  'Create an instance of the list object. This syntax is more efficient than when only the new instruction is used. Otherwise, before execution, Visual Basic will test each statement containing list for the value Nothing. '
list.StartUpdates RT_MODE_TIME
list.Mode = "FRQ:10s"
```

Source

Name of the data source as a String.

Arguments

None

Error handling

Depends on the value of the AdxRtList DebugLevel general property, as follows:

Value	If the source status is	Then
`RT_DEBUG_NO`	already known as invalid or down	an exception is raised
`RT_DEBUG_NO`	unknown	then the process continues without any warning or error
`RT_DEBUG_IMMEDIATE`	invalid	an exception is raised before setting a new property value.
	• unknown • down	an exception is raised.
	already known as invalid or down	an exception is raised
	unknown	the process continues without any warning or error

VBA sample

```
Public list as AdxRtList                    'Declares as a global variable'
Set list = CreateAdxRtList                  'Creates an instance of the list object. This syntax is more efficient only when the New command is used. Otherwise, before execution, Visual Basic tests each statement containing list for the value Nothing.'
list.Source = "IDN"                         'Sets the source name to "IDN"'
```

See also

"AdxRtChain Source" on page 89
"AdxRtChain SourceStatus" on page 89
"AdxRtContribute Source" on page 84
"AdxRtHistory Source" on page 96
"AdxRtList SourceStatus" on page 66
"RT_SourceStatus" on page 103

This property returns a value from the RT_SourceStatus enumeration. This value reflects the status of the source defined by the AdxRtList Source property of the AdxRtList object.

Arguments

None

Return value

Returns the status of the source of the list object.

Error handling

If the AdxRtList Source is not set (i.e. the source is in the state RT_SOURCE_NOT_SET, see RT_SourceStatus), an error exception is thrown.

VBA sample

```
Public list as
AdxRtList

Set list =
CreateAdxRtList

list.Source =
"IDN"

Status =
list.SourceStatus
```

'Sets `Status` to the status of the `"IDN"` source; this might be `RT_SOURCE_UP` or `RT_SOURCE_DOWN`, for example'

See also

"AdxRtChain Source" on page 89
"AdxRtChain SourceStatus" on page 89
"AdxRtContribute Source" on page 84
"AdxRtHistory Source" on page 96
"AdxRtList Source" on page 66
"RT_SourceStatus" on page 103

AdxRtList UserTag

Allows information to be associated with an item or with a single field of an item in the list. The value is held in the association as a Variant.

Different user tag values associated with items and fields can be consulted using AdxRtList ListItems and AdxRtList ListFields. The tag associated with an item is also provided as a parameter to the AdxRtList OnUpdate event callback function.

Note that user tag values related to fields must be integer values (of type Long).

Arguments

`ItemName`	The name of the required item as a String.
`FieldName`	The name of the field to which the user tag applies. If `FieldName` has the form `""` (the empty string) or `"*"` then the user tag is applied to the whole item, otherwise it concerns the field itself.

Return value

When used to retrieve the user tag, a Variant value is returned.

Error handling

An error exception may be thrown in the following cases:

Condition	Exception thrown
AdxRtList Source is not set (i.e. the source is in the state RT_SourceStatus)	Error
Either the `ItemName` or the `FieldName` is not registered in the AdxRtList object (using AdxRtList RegisterItems)	Invalid argument
The AdxRtList DebugLevel general property is `RT_DEBUG_NO`, and either the ItemName or the FieldName is already known to be invalid	Invalid argument
The AdxRtList DebugLevel general property is `RT_DEBUG_IMMEDIATE`. The program waits for a callback to determine the validity of either `ItemName` or `FieldName`. If either is invalid, an exception is thrown.	Invalid argument

ⓘ This approach is inherently slow.

VBA sample

```
Public list as
AdxRtList

Set list =
CreateAdxRtList

list.UserTag
("TRI.TO", "")
= "Port:cld"        'Associates the String value "Port:cld" with the item "TRI.TO" in list's item list.
                    When the AdxRtList OnUpdate event associated with this item is handled, this string
                    value is passed as the UserTag parameter to the callback function.'

list.UserTag
("DEM=", "BID")
= 100               'Associates the integer value 100 with the item "DEM="
```

AdxRtList Value

This property returns a Variant containing the current value of the field specified by ”FieldName” which belongs to the item specified by ”ItemName”.

If the data is not available or one of the parameters is invalid, an empty Variant is returned.

Arguments

`Field`	The name or numeric identifier of the field whose value is to be retrieved.
`ItemName`	The name of the item as a String.

Return value

Returns the value of the identified field or an empty Variant if the value is unknown or invalid.

Error handling

If the item or the field is not registered or the field is not in the field dictionary, an exception is thrown.

VBA sample

```
Public list as AdxRtList
Set list = CreateAdxRtList
Dim ItemList(0 To 1) as Variant
ItemList(0) = "EUR="
ItemList(1) = "JPY="
Dim FieldList(0 To 1) as
Variant
FieldList(0) = "BID"
FieldList(1) = "ASK"
Dim Data As Variant                    'The returned value can be either a String or a numeric value. '
list.RegisterItems ItemList,
FieldList
Data = list.Value("EUR=",              'The latest value of the "BID" field of the item "EUR=" is
"BID")                                 stored in Data. '
```

AdxRtList Methods

AdxRtList CloseAllLinks

This method stops the delivery of all updates for items registered in the AdxRtList object. It closes down the underlying data stream by informing the data source that no more updates would be necessary for those items.

Arguments

None

Error handling

If AdxRtList Source has not been set (i.e. the source is in the state RT_SOURCE_NOT_SET, see "RT_SourceStatus" on page 103), an error exception is thrown.

VBA sample

```
Public list as
AdxRtList
Set list =
CreateAdxRtList
list.CloseAllLinks 'Stops the delivery of data for all items registered in the list list and tells the
                   source that they are no longer required.'
```

See also

"AdxRtList StartUpdates" on page 72
"AdxRtList StopUpdates" on page 73
"RT_RunMode" on page 104

AdxRtList IsRegisteredField

This property returns a Boolean value of `True` if the given item of the AdxRtList object contains the designated field. Otherwise it returns `False`.

Arguments

`ItemName`	The name of the item as a String
`FieldName`	The name of the field as a String

Return value

`True` if the field was found, `False` otherwise.

VBA sample

```
Public list as AdxRtList
Set list = CreateAdxRtList
Dim ItemList(0 To 1) as Variant
ItemList(0) = "EUR="
ItemList(1) = "JPY="
Dim FieldList(0 To 1) as Variant
FieldList(0) = "BID"
FieldList(1) = "ASK"
list.RegisterItems ItemList, FieldList
TestBid = list.IsRegisteredField("EUR=", "BID")                'Sets TestBid to True
TestDName = list.IsRegisteredField("JPY=", "DISPLAY_           'Sets TestDName to
NAME")                                                         False'
```

AdxRtList IsRegisteredItem

This property returns a Boolean value of `True` if the designated item is registered in the AdxRtList object. Otherwise it returns `False`.

Arguments

`ItemName`	The name of the item as a String.

Return value

`True` if the item was found, `False` otherwise.

VBA sample

```
Public list as AdxRtList
Set list = CreateAdxRtList
Dim ItemList(0 To 1) as Variant
ItemList(0) = "EUR="
ItemList(1) = "JPY="
list.RegisterItems ItemList, FieldList
TestEUR = list.IsRegisteredItem("EUR=")          'Sets TestEUR to True'
TestCHF = list.IsRegisteredItem("CHF=")          'Sets TestCHF to False'
```

AdxRtList RegisterItems

This method adds a list of items with a corresponding list of associated fields to the AdxRtList object. Registering the items allows for retrieval of item data from the object data source (defined by the AdxRtList Source property).

Arguments

`ItemList`	An array of Variant containing one or more item names.
`FieldList`	An array of Variant containing one or more item names. It is possible, by specifying the single value "*", to request all fields present in the items. (This functionality depends on the system used to provide access to the data sources.)

Return value

If successful, this method returns `RT_OK`, otherwise it returns the error status `RT_ERR`. An error means that the `FieldList` is not correctly set and that at least one field is undefined (has a field status of `RT_FIELD_UNDEFINED`, see "RT_FieldStatus" on page 102).

Error handling

Depends on the value of the AdxRtList DebugLevel general property, as follows:

RT_DEBUG_NO	If one of the items in the ItemList or one of the fields in the FieldList is already known to be invalid, an exception is thrown.
RT_DEBUG_IMMEDIATE	On direct API platforms, for each item in the ItemList, an initial event callback is used to determine whether the item or any of the requested fields within it is invalid. If so, an exception is thrown accordingly. It is important to note that in this mode the registration process can be very slow.

VBA sample

```
Public list as
AdxRtList

Set list =
CreateAdxRtList

Dim ItemList(0 To 1) as
Variant

ItemList(0) = "EUR="

ItemList(1) = "JPY="

Dim FieldList(0 To 1)
as Variant

FieldList(0) = "BID"

FieldList(1) = "ASK"

list.RegisterItems
ItemList, FieldList      'Adds the items "EUR=" and "JPY=" to the list list. For each of these
                         items, the fields "BID" and "ASK" are required
```

See also

"AdxRtList CloseAllLinks" on page 69
"AdxRtList StartUpdates" on page 72
"AdxRtList StopUpdates" on page 73
"RT_RunMode" on page 104

AdxRtList StartUpdates

This method starts the real time update of data for the items registered using AdxRtList RegisterItems in the AdxRtList object. It also establishes the update mode.

Arguments

Mode	A value from the RT_RunMode enumeration.

Error handling

If Mode contains an invalid value, an error exception is thrown.

If the "AdxRtList Source" on page 66 is not set (i.e. the source is in the state RT_SOURCE_NOT_SET, see "RT_SourceStatus" on page 103), an error exception is thrown.

VBA sample

```
Public list as
AdxRtList
Set list =
CreateAdxRtList
list.StartUpdates
 RT_MODE_IMAGE      'Requests all items registered in the list list from its source. Once a data image
                    is received for each of the objects (or a time-out occurs), no more data will be
                    received. This supplies a snapshot of the items.'
```

See also

"AdxRtList CloseAllLinks" on page 69
"AdxRtList StopUpdates" on page 73
"RT_RunMode" on page 104

AdxRtList StopUpdates

This method stops the delivery of all updates of items registered in the AdxRtList object to the program but without closing the underlying data stream unless the AdxRtList StartUpdates mode was RT_RunMode or the AdxRtList object is in the RT_LIST_LINKS_CLOSED state (see "RT_ListStatus" on page 103). This allows updates to be stopped momentarily. To resume updates, the AdxRtList StartUpdates method should be called again.

Arguments

None

Error handling

If the AdxRtList Source is not set (i.e. the source is in the state RT_SOURCE_NOT_SET, see RT_SourceStatus), an error exception is thrown.

VBA sample

```
Public list as AdxRtList
Set list = CreateAdxRtList
list.StopUpdates            'Stops the delivery of data for all items registered in the list list.'
```

See also

"AdxRtList CloseAllLinks" on page 69
"AdxRtList StartUpdates" on page 72
"RT_RunMode" on page 104

AdxRtList UnregisterAllItems

This method removes all items from the AdxRtList object which were previously registered with AdxRtList RegisterItems. If the object is receiving events (data is requested using the AdxRtList StartUpdates method), no more information is received by the object, as if the AdxRtList StopUpdates method were called.

Arguments

None

Error handling

If the AdxRtList Source is not set (i.e. the source is in the state `RT_SOURCE_NOT_SET`, see "RT_SourceStatus" on page 103), an error exception is thrown.

VBA sample

```
Public list as AdxRtList
Set list = CreateAdxRtList
list.UnregisterAllItems                'Removes all registered items from the list list. '
ItemCount = list.ItemListCount         'This sets ItemCount to 0.'
```

AdxRtList UnregisterFields

This method removes the fields identified by `FieldList` from the items in `ItemList`, which should have been registered in the AdxRtList object earlier using AdxRtList RegisterItems. If by using this method all registered fields are removed for an item, this does not unregister the item itself.

Arguments

`ItemList`	A list of one or more names as a Variant, identifying the items whose fields should be removed.
`FieldList`	A list of one or more names as a Variant, identifying the fields to remove. These must be fields which were named explicitly using the AdxRtList RegisterItems`FieldList` parameter.

Error handling

If one of the item names in `ItemList` or one of the field names in `FieldList` is not registered previously in the list object, an error exception is thrown.

If the AdxRtList Source is not set (i.e. the source is in the state `RT_SOURCE_NOT_SET`, see "RT_SourceStatus" on page 103), an error exception is thrown.

VBA sample

```
Public list as AdxRtList
Set list = CreateAdxRtList
Dim ItemList as Variant
ItemList = "EUR="
Dim FieldList as Variant
FieldList = "ASK"
list.UnregisterFields ItemList,        'Removes the "ASK" field from the item "EUR=" of
FieldList                               the list.'
```

AdxRtList UnregisterItems

This method removes a list of items from the AdxRtList object which were previously registered with AdxRtList RegisterItems. If the object is receiving events (data is requested using the AdxRtList StartUpdates method), no more information is received by the object concerning these items.

Arguments

`ItemList`	A list of one or more names as a `Variant` identifying the items to remove from the list.

Error handling

If one of the item names in `ItemList` is not previously registered in the list object, an error exception is thrown.

If the AdxRtList Source is not set (i.e. the source is in the state `RT_SOURCE_NOT_SET`, see "RT_SourceStatus" on page 103), an error exception is thrown.

VBA sample

```
Public list as AdxRtList
Set list = CreateAdxRtList
Dim ItemList as Variant
ItemList = Array("EUR=", "JPY=")
list.UnregisterItems ItemList          'Removes the item "EUR=" from the list list. '
```

AdxRtList Events

- "AdxRtList OnImage" on page 75
- "AdxRtList OnStatusChange" on page 76
- "AdxRtList OnTime" on page 76
- "AdxRtList OnUpdate" on page 77

AdxRtList OnImage

This event is triggered once a data image is received for each of the registered items in AdxRtList that is receiving data following a call to AdxRtList StartUpdates using the `RT_MODE_IMAGE` mode (see "RT_RunMode" on page 104). It is also called if a time-out occurs (set using the `TIME-OUT` attribute of the `Mode` property, see "AdxRtList Mode" on page 64) before all data images could be received.

User code should provide a callback function to handle these events. In this function, the "AdxRtList ListItems" on page 63 method can be used to see which items have a data image associated with them. For each of these, the "AdxRtList ListFields" on page 61 method can be called to obtain the values of its fields.

Arguments

`DataStatus`	The status of the registered items, a value from the RT_DataStatus enumeration. The status value is `RT_DS_FULL` if data is received from the source for all registered items, or `RT_DS_PARTIAL` if at least one of the items still has no associated data at the end of the time-out period.

See also

"AdxRtList CloseAllLinks" on page 69
"AdxRtList StartUpdates" on page 72
"AdxRtList StopUpdates" on page 73
"RT_RunMode" on page 104

AdxRtList OnStatusChange

This event is generated every time a change occurs that affects the status of the AdxRtList object as a whole. Such events are:

- The AdxRtList Source property is changed.
- The RT_RunMode changes (through a call to the AdxRtList StartUpdates method).
- The real time data streams are stopped (through external events) or closed (through a call to the AdxRtList CloseAllLinks method).
- The AdxRtList object goes from empty to non-empty.

User code should supply a callback function to handle these events within the program.

Arguments

`ListStatus`	The current status of the AdxRtList object. This is a value from the RT_ListStatus enumeration.
`SourceStatus`	The current status of the data source (identified by the AdxRtList Source property). This is a value from the RT_SourceStatus enumeration.
`RunMode`	The current update mode, usually changed through the AdxRtList StartUpdates method. This is a value from the RT_RunMode enumeration.

AdxRtList OnTime

This event is generated regularly by a timer at a frequency determined by the `FRQ` attribute parameter (see "AdxRtList Mode" on page 64) on behalf of the AdxRtList object that is receiving data following a call to AdxRtList StartUpdates using `RT_MODE_ONTIME` (see RT_RunMode mode).

User code should provide a callback function to handle these events. In this function, the AdxRtList method can be used to see which items have changed since the last `OnTime` event. For each of these, the AdxRtList ListFields method can be called to obtain the values of the updated fields.

Arguments

None

See also

"AdxRtList CloseAllLinks" on page 69
"AdxRtList StartUpdates" on page 72
"AdxRtList StopUpdates" on page 73

"RT_RunMode" on page 104

AdxRtList OnUpdate

When AdxRtList StartUpdates is called with either the `RT_MODE_ONUPDATE` (see "RT_RunMode" on page 104) or `RT_MODE_ONTIME_IF_UPDATED` mode (see "RT_RunMode" on page 104), events of this type are sent to the user application when data updates from the source are received for any of the items registered by the object.

An event is generated for each item whose data is updated by the data source, either as soon as the change is received (when `RT_MODE_ONUPDATE` is used, see "RT_RunMode" on page 104), or if the change occurred during the period between regular checks controlled by a timer (when `RT_MODE_ONTIME_IF_UPDATED` is used, see "RT_RunMode" on page 104).

User code should provide a callback function to handle these events. In this function, the AdxRtList ListFields method can be called to determine which fields in the item have changed, and what the new field values are.

Arguments

`ItemName`	The name of one of the items registered in the list object. The update event is associated with this item.
`UserTag`	The user tag associated with the item, if one was supplied using the AdxRtList UserTag method. This parameter should be treated as a constant; it must not be changed by the application code.
`ItemStatus`	The status of the item, a value from the AdxRtList ItemStatus enumeration.

AdxRtList IListEvents Interface

A user application can use the `IListEvents` interface as an alternative to the standard event (connection point) interface associated with an object of type AdxRtList. To use this alternative interface, the user application must associate an object which implements the IListEvents interface with the AdxRtList object or objects. This is done using the AdxRtList ListAdvise method. The association can be broken by calling the AdxRtList ListUnadvise method.

Once an association of this type is established, the various `AdxRtList Events` delivered for the AdxRtList object will cause the corresponding methods of the `IListEvents` object to be called.

Using this interface allows methods to be dispatched using a vtable mechanism. This is a faster process than the connection point mechanism.

- "AdxRtList ListAdvise" on page 78
- "AdxRtList ListUnadvise" on page 79
- "AdxRtList OnImage" on page 79
- "AdxRtList OnStatusChange" on page 79
- "AdxRtList OnTime" on page 80
- "AdxRtList OnUpdate" on page 80

VBA sample

A Visual Basic implementation of an object providing the `IListEvents` interface involves using a class module. This will look something like the following:

```
Implements
IListEvents
```

```
Dim list as AdxRtList                'This declares that the object implemented by the class module contains a list object...'
Set list = CreateAdxRtList           '... which is created here. '
list.ListAdvise Me                   'This associates the object with the list item it knows about, list. In other words, list 's events will be sent to the object containing it. '
list.StartUpdates RT_MODE_ONUPDATE   'Asks for updates for items registered by the list. '
```

The following event callback for the interface can be created by the Visual Basic editor.

```
Sub IListEvents_OnUpdate(ByVal MyList as AdxRtxLibCtl.IAdxRtList, _
ItemName as String,
UserTag as Long,
ItemStatus as RT_ItemStatus)
Dim MyArray as Variant
MyArray = MyList.ListItems(RT_IRV_UPDATED, RT_ICV_USERTAG)   'The objects list (a property of the class module's object) and MyList (received as a parameter to this function) are in fact the same. '
End Sub
```

See also

- "AdxRtList CloseAllLinks" on page 69
- "AdxRtList StartUpdates" on page 72
- "AdxRtList StopUpdates" on page 73
- "RT_RunMode" on page 104

AdxRtList ListAdvise

This method allows the user application to receive events for the AdxRtList object though an alternative interface. The event callback functions are methods of an object implementing the AdxRtList IListEvents Interface.

When AdxRtList StartUpdates method is called, the events will be delivered to the methods of the interface object.

Arguments

pListEvents	A pointer to an object implementing the AdxRtList IListEvents Interface. This object need not be unique for all AdxRtList objects.

See also

"AdxRtList IListEvents Interface" on page 77

"AdxRtList ListUnadvise" on page 79

"AdxRtList OnImage" on page 79

"AdxRtList OnUpdate" on page 80

"AdxRtList OnTime" on page 80

"AdxRtList OnStatusChange" on page 79

AdxRtList ListUnadvise

This method detaches the AdxRtList object from an object implementing the AdxRtList IListEvents Interface.

Arguments

None

See also

"AdxRtList IListEvents Interface" on page 77
"AdxRtList ListAdvise" on page 78
"AdxRtList OnImage" on page 79
"AdxRtList OnUpdate" on page 80
"AdxRtList OnTime" on page 80
"AdxRtList OnStatusChange" on page 79

AdxRtList OnImage

The OnImage event causes a call to this method in the object providing the AdxRtList IListEvents Interface for the AdxRtList object.

The two objects are linked through a call to the AdxRtList ListAdvise method.

Arguments

`MyList`	A pointer to the `IAdxRtList` interface of the AdxRtList object for which the event was delivered.
`DataStatus`	The status of the registered items, a value from the RT_DataStatus enumeration. This will have the value RT_DS_FULL if data is received from the source for all registered items, or RT_DS_PARTIAL if at least one of the items still has no associated data at the end of the time-out period.

See also

"AdxRtList IListEvents Interface" on page 77
"AdxRtList ListAdvise" on page 78
"AdxRtList ListUnadvise" on page 79
"AdxRtList OnUpdate" on page 80
"AdxRtList OnTime" on page 80
"AdxRtList OnStatusChange" on page 79

AdxRtList OnStatusChange

The `OnStatusChange` event causes a call to this method in the object providing the AdxRtList IListEvents Interface for the AdxRtList object.

The two objects were linked through a call to the AdxRtList ListAdvise method of the AdxRtList object.

Arguments

`ListStatus`	The current status of the AdxRtList object. This is a value from the AdxRtList ListStatus enumeration.
`MyList`	A pointer to the `IAdxRtList` interface of the AdxRtList object for which the event was delivered.
`SourceStatus`	The current status of the data source (identified by the AdxRtList Source property). This is a value from the RT_SourceStatus enumeration.
`RunMode`	The current update mode, usually changed through the AdxRtList StartUpdates method. This is a value from the RT_RunMode enumeration.

See also

"AdxRtList IListEvents Interface" on page 77
"AdxRtList ListAdvise" on page 78
"AdxRtList ListUnadvise" on page 79
"AdxRtList OnImage" on page 79
"AdxRtList OnUpdate" on page 80
"AdxRtList OnTime" on page 80

AdxRtList OnTime

The `OnTime` event causes a call to this method in the object providing the AdxRtList IListEvents Interface for the AdxRtList object.

The two objects were linked through a call to the AdxRtList ListAdvise method.

Arguments

`MyList`	A pointer to the `IAdxRtList` interface of the AdxRtList object for which the event was delivered.

See also

"AdxRtList IListEvents Interface" on page 77
"AdxRtList ListAdvise" on page 78
"AdxRtList ListUnadvise" on page 79
"AdxRtList OnImage" on page 79
"AdxRtList OnUpdate" on page 80
"AdxRtList OnStatusChange" on page 79

AdxRtList OnUpdate

The `OnUpdate` event causes a call to this method in the object providing the AdxRtList IListEvents Interface for the AdxRtList object.

The two objects are linked through a call to the AdxRtList ListAdvise method.

Arguments

`MyList`	A pointer to the `IAdxRtList` interface of the AdxRtList object for which the event was delivered.

`ItemName`	The name of one of the items registered in the list object identified by `MyList`. The update event is associated with this item. This parameter should be treated as constant: it must not be changed by the application code.
`UserTag`	The user tag associated with the item, if one was supplied using the AdxRtList UserTag method. This `Variant` value should be treated as constant.
`ItemStatus`	The status of the item, a value from the RT_ItemStatus enumeration.

See also

"AdxRtList IListEvents Interface" on page 77
"AdxRtList ListAdvise" on page 78
"AdxRtList ListUnadvise" on page 79
"AdxRtList OnImage" on page 79
"AdxRtList OnTime" on page 80
"AdxRtList OnStatusChange" on page 79

AdxRtContribute

AdxRtContribute Properties

AdxRtContribute Attribute

Allows the consultation of various settings applied to the object. These are the settings that can be changed by assigning values to the AdxRtContribute Mode property.

Arguments

`AttrID` An identification of the attribute, passed as a Variant. This can be either an identifying code taken from the AdxAttrRtContribute enumeration or a string containing the keyword used to set the attribute using the AdxRtContribute Mode property.

Return value

The value of the requested attribute of the type `Variant`.

AdxRtContribute ErrorCode

This property retrieves the error code of the latest error encountered by the object as an integer value. If the `ErrorMode` property of the object is set to `NO_EXCEPTION` (see "AdxErrorMode" on page 106), this value must be consulted by the client application to determine whether an error has occurred.

Arguments

None

AdxRtContribute ErrorMode

This property sets the error mode for the object. When an error is detected, this property is consulted to see what action should be taken by the object to inform the user of the application. It must be assigned a value from the AdxErrorMode enumeration.

Default value

By default the error mode is set to `EXCEPTION` (see "AdxErrorMode" on page 106).

Arguments

None

AdxRtContribute ErrorString

This property retrieves a string describing the latest error encountered by the object. If the `ErrorMode` property of the object is set to NO_EXCEPTION (see "AdxErrorMode" on page 106), this value can be consulted by the client application when the `ErrorCode` property indicates that an error has occurred.

Arguments

None

AdxRtContribute ItemName

This property provides the name of the item as a string which identifies the item to which to contribute. It must be set in order to be able to perform the contribution.

Arguments

None

AdxRtContribute Mode

This property describes various behavioral aspects of the AdxRtContribute object. In particular, it determines how the contribution object will deliver data update events to the client application. The allowed parameters are:

`POS`	The position in a line at which the value should be written, for contribution to a page.
`POS:i`	the (formatted) field value is to be pasted over the contents of the destination line starting at the i^{th} character position.

Arguments

None

AdxRtContribute RunStatus

This property allows the application to know whether the object is currently performing a contribution.

Arguments

None

Return value

This function returns one of the values listed in the RT_RunStatus enumeration.

AdxRtContribute Source

Name of the data source as a string to which the contribution request is to be addressed.

Arguments

None

AdxRtContribute Methods

AdxRtContribute Contribute

This method contributes multiple field values for the item indicated by the AdxRtContribute ItemName property.

Arguments

`FieldNameArray`	Variant, typically an array of field names as `Strings`.
`FieldValueArray`	Variant, typically an array of field values of type `Variant`. There should be one field value for each field name in `FieldNameArray`.

Return value

Returns True if the contribution has been sent successfully, False otherwise.

VBA sample

```
Dim contrib as AdxRtContribute
Set contrib = CreateAdxRtContribute
contrib.Source = "DDS"
contrib.ItemName = "EUR="
ret = contrib.Contribute("BID",                    'Contribute a "BID" field value for the item "EUR="
"1.0512")                                          directly. '
```

```
Dim FieldNameArray(0 to 1) as Variant
Dim FieldValueArray(0 to 1) as Variant
contrib.ItemName = "FRF="
FieldNameArray(0) = "BID"
FieldNameArray(1) = "ASK"
FieldValueArray(0) = 5.8910
FieldValueArray(1) = 5.8950
ret = RtObject.Contribute                    'Contribute "BID" and "ASK" field values for the
FieldNameArray, FieldValueArray              item "FRF=" using arrays. '
```

AdxRtChain

AdxRtChain Properties

AdxRtChain Attribute

Allows the consultation of various settings applied to the object. These are the settings that can be changed by assigning values to the AdxRtChain Mode property.

Arguments

`AttrID` An identification of the attribute, passed as a Variant. This can be either an identifying code taken from the AdxRtChain enumeration or a string containing the keyword used to set the attribute using the AdxRtChain Mode property.

Return value

The value of the requested attribute of the type Variant.

AdxRtChain Data

This property returns the contents of the chain, as a Variant array.

Arguments

None

Return value

Returns a one-dimensional array of Variant. The elements contain the names of the items making up the chain as strings. The AdxRtChain Mode property determines various presentational options.

VBA sample

```
Dim chain as AdxRtChain
Set chain = CreateAdxRtChain
chain.Source = "IDN"                    'Set the chain item's source.'
chain.ItemName = ".FCHI"                'Set the chain item's name.'
chain.Mode = "IGNE:YES"                 'Remove empty entries.'
Chain.RequestChain
ret = chain.Data                        'Retrieve the list of items in the ".FCHI" chain.'
```

AdxRtChain DataStatus

This property returns the status of the chain data received from the source.

Arguments

None

Return value

This property returns a value taken from the RT_DataStatus enumeration.

AdxRtChain ErrorCode

This property retrieves the error code of the latest error encountered by the object as an integer value. If the `ErrorMode` property of the object is set to `NO_EXCEPTION` (see AdxErrorMode), this value must be consulted by the client application to determine whether an error has occurred.

Arguments

None

AdxRtChain ErrorMode

This property sets the error mode for the object. When an error is detected, this property is consulted to see what action should be taken by the object to inform the user of the application. It must be assigned a value from the AdxErrorMode enumeration.

Default value

By default the error mode is set to EXCEPTION (see "AdxErrorMode" on page 106).

Arguments

None

AdxRtChain ErrorString

This property retrieves a string describing the latest error encountered by the object. If the `ErrorMode` property of the object is set to `NO_EXCEPTION` (see "AdxErrorMode" on page 106), this value can be consulted by the client application when the `ErrorCode` property indicates that an error has occurred.

Arguments

None

AdxRtChain ItemName

This property provides the name of the item as a string which identifies the chain. It must be set in order to enable chain request.

Arguments

None

AdxRtChain Mode

This property describes how the AdxRtChain Data property is to present its results. It is a string which contains a sequence of parameter settings. The parameters are:

Parameter	Description	Possible value
`IGNE`	The *Ignore Empty* parameter determines whether empty elements of the returned chain contents are removed.	`IGNE:YES`: remove empty entries from the array `IGNE:NO`: leave empty entries in the array
`LAY`	The *Layout* parameter determines how the elements are arranged in the returned array.	`LAY:HOR or LAY:H`: return items in a horizontal layout `LAY:VER or LAY:V`: return items in a vertical layout
`LIVE`	The *Live* parameter determines whether the chain should continue to be supplied with updates once the initial chain data have been fully received. This allows an application to decide whether to track changes to the contents of the chain coming from the data source.	`LIVE:YES`: continue to receive updates once the chain has been received completely `LIVE:NO`: supply no more updates once the chain data is complete (the default)
`RET`	The *Return* parameter allows the size of the returned array to be controlled.	`RET:An`: return the first n entries of the array as an array

Parameter	Description	Possible value
SKIP	The *Skip* parameter allows various entries in the chain to be skipped before returning a reduced array. Entries are numbered from 1. By default no elements are skipped. Often the first few entries of a chain do not contain item names.	SKIP:n: remove the n^{th} entry from the array SKIP:i-j: remove all entries from the i^{th} to the j^{th} (inclusive) from the array SKIP:i-j,k: different entries or ranges to skip can be combined in a comma-separated list
UWC	The Update When Completed parameter determines whether update events will be signaled to the application through the "AdxRtChain OnUpdate" on page 91 event callback while the different constituent parts of the chain are being retrieved. By default this parameter is active (UWC:YES).	• UWC:NO: signal an update event for the retrieval of each part of the chain • UWC:YES: wait until the chain has been retrieved completely before generating an update event

Arguments

None

AdxRtChain RunStatus

This property allows the application to know whether the object is currently retrieving chain data.

Arguments

None

Return value

This function returns one of the values listed in the RT_RunStatus enumeration.

AdxRtChain Source

The name of the data source as a string. It must be set in order to enable chain request.

Arguments

None

AdxRtChain SourceStatus

This property returns a value from the RT_SourceStatus enumeration. This value reflects the status of the source defined by the AdxRtChain Source property of the AdxRtChain object.

Arguments

None

Return value

Returns the status of the source of the list object.

Error handling

If the AdxRtChain Source has not been set (i.e. the source is in the state `RT_SOURCE_NOT_SET`, see "RT_SourceStatus" on page 103) an error exception is thrown.

```
Public chain as
AdxRtChain

Set chain =
CreateAdxRtChain

chain.Source =
"IDN"

Status =                'Sets Status to the status of the "IDN" source; this might be RT_SOURCE_UP
chain.SourceStatus       or RT_SOURCE_DOWN, for example.'
```

See also

"AdxRtChain Source" on page 89
"AdxRtContribute Source" on page 84
"AdxRtHistory Source" on page 96
"AdxRtList Source" on page 66
"AdxRtList SourceStatus" on page 66
"RT_SourceStatus" on page 103

AdxRtChain Methods

AdxRtChain RequestChain

This method retrieves the data for the chain specified by the AdxRtChain Source and AdxRtChain ItemName parameters. This action occurs asynchronously; the arrival of chain data, or an indication of a request error, causes the AdxRtChain OnUpdate event to occur.

Arguments

None

VBA sample

```
Dim chain as AdxRtChain
Set chain = CreateAdxRtChain
chain.Source = "IDN"                    'Set the chain item's source.'
chain.ItemName = ".FCHI"                'Set the chain item's name.'
chain.Mode = "IGNE:YES"                 'Remove empty entries.'
chain.RequestChain                      'Ask for the chain data.'
```

AdxRtChain Events

AdxRtChain OnStatusChange

This callback is called every time a change occurs to the status of the data source specified by the AdxRtChain Source property while there is an outstanding AdxRtChain RequestChain request.

Arguments

`SourceStatus`	The current status of the data source (identified by the AdxRtChain Source property). This is a value from the RT_SourceStatus enumeration.

See also

"AdxRtList CloseAllLinks" on page 69

"AdxRtList StartUpdates" on page 72

"AdxRtList StopUpdates" on page 73

"RT_RunMode" on page 104

AdxRtChain OnUpdate

This callback handles updates to the requested data. If the Update When Completed attribute is in effect (see the `"UWC"` keyword of the AdxRtChain Mode property), this handling will occur only once all chain data is retrieved, or a request error occurs. Otherwise, the event may be called a number of times before all data have arrives.

If the Live attribute is active (see the "LIVE" keyword of the AdxRtChain Mode property), updates may continue to arrive following the completion of the chain data, as the source sends changes to the chain contents.

Arguments

`DataStatus`	The status of the data retrieved by the request. This is a value taken from the RT_DataStatus enumeration.

Important! `AdxRtHistory` relies on TS1 data, whose distribution will stop in mid-2016. Do not use this API for new developments.

AdxRtHistory

AdxRtHistory Properties

AdxRtHistory Attribute

Allows the consultation of various settings applying to the object. These are the settings that can be changed for the object by assigning values to the AdxRtHistory ErrorMode property.

Arguments

`AttrID` An identification of the attribute, passed as a Variant. This can be either an identifying code taken from the AdxAttrRtHistory enumeration or a string containing the keyword used to set the attribute using the AdxRtHistory ErrorMode property.

Return value

The value of the requested attribute of type `Variant`.

AdxRtHistory Data

This property returns the historical data as a `Variant` array.

Arguments

None

Important! `AdxRtHistory` relies on TS1 data, whose distribution will stop in mid-2016. Do not use this API for new developments.

Return value

Returns a two-dimensional array of `Variant`. The elements contain values for the fields listed in AdxRtHistory ExistingFields for each date entry retrieved from the time series. The AdxRtHistory Mode property determines various presentational and selection filtering options.

AdxRtHistory DataStatus

This property returns the status of the historical data received from the source.

Arguments

None

Return value

This property returns a value taken from the RT_DataStatus enumeration.

AdxRtHistory ErrorCode

This property retrieves the error code of the latest error encountered by the object as an integer value. If the `ErrorMode` property of the object is set to `NO_EXCEPTION` (see AdxErrorMode), this value must be consulted by the client application to determine whether an error has occurred.

Arguments

None

AdxRtHistory ErrorMode

This property sets the error mode for the object. When an error is detected, this property is consulted to see what action should be taken by the object to inform the user of the application. It must be assigned a value from the AdxErrorMode enumeration.

Default value

By default, the error mode is set to `NO_EXCEPTION`; see "AdxErrorMode" on page 106.

Arguments

None

AdxRtHistory ErrorString

This property retrieves a string describing the latest error encountered by the object. If the `ErrorMode` property of the object is set to `NO_EXCEPTION` (see "AdxErrorMode" on page 106), this value can be

Important! `AdxRtHistory` relies on TS1 data, whose distribution will stop in mid-2016. Do not use this API for new developments.

consulted by the client application when the `ErrorCode` property indicates that an error has occurred.

Arguments

None

AdxRtHistory ExistingFields

This property indicates which fields have been supplied in the returned historical data entries.

Arguments

None

Return value

Returns a one-dimensional array of Variant. The elements contain the names of the fields in each date entry of the AdxRtHistory Data property, as returned by the time series data source.

AdxRtHistory ItemName

This property provides the name of the item for which the historical time series is to be retrieved.

Arguments

None

AdxRtHistory Mode

This property determines how the AdxRtHistory Data property is to present its results. It is a string which contains a sequence of parameter settings. It also allows the definition of the range of dates over which historical data is to be retrieved. The parameters are:

Parameter	Description	Possible Value
`END`	The *End Date* parameter specifies the last date for which historical data should be retrieved. By default data will be retrieved up to the current date.	`END:ddmmmyy`: sets an end date of ddmmmyy
`HEADER`	The *Header* attribute determines whether a header row containing field names should be returned along with the data. If the keyword `HEADER` is present on its own, it is treated as `HEADER:YES`.	`HEADER:YES`: include a header row `HEADER:NO`: do not include a header row (default)

Important! `AdxRtHistory` relies on TS1 data, whose distribution will stop in mid-2016. Do not use this API for new developments.

Parameter	Description	Possible Value
`FRQ`	The *Frequency* parameter establishes the predefined time-series to consult.	`FRQ:D`: Fetch daily data (default) `FRQ:W:` Fetch weekly data `FRQ:M`: Fetch monthly data `FRQ:other`: Fetch data of the frequency specified by the letter in other. This frequency must be supported by the source
`LAY`	The *Layout* attribute determines how the elements are to be arranged in the returned array.	`LAY:HOR or LAY:H`: returns items in a horizontal layout `LAY:VER or LAY:V`: returns items in a vertical layout
`NBEVENTS`	The *Number of Events* parameter limits the number of historical events that will be retrieved. The application of the `NBEVENTS` limit involves counting date entries with null data when the `NULL:NA` or `NULL:REPEAT` condition is used; these entries are ignored if `NULL:SKIP` is used. The default value depends on the frequency setting.	`NBEVENTS:n`: return data for no more than n events `NBEVENTS:25`: with a daily frequency (`FRQ:D`) `NBEVENTS:52`: with a weekly frequency (`FRQ:W`) `NBEVENTS:12`:with a monthly frequency (`FRQ:M`) `NBEVENTS:25`: with any other frequency (`FRQ:other`)
`EVENTS`	Exact number of historical data retrieved {`i` with `i` as integer}	`EVENTS:i`: to retrieve i data points (see the NBEVENTS keyword and the Note section below.)
`NULL`	The *Null* attribute dictates how empty entries should be handled. The `NULL` keyword defines the action taken only where date entries with null data are encountered. Dates before the start date or following the end date of the time series are always ignored.	`NULL:NA`: provide the String value `"#N/A ND"` instead of null values (default) `NULL:REPEAT`: copy earlier valid values to this entry, if any `NULL:SKIP:` drop the entry in the returned array
`RET`	The *Return* parameter allows the size of the returned array to be controlled.	`RET:An`: return the first n entries of the array as an array

Important! `AdxRtHistory` relies on TS1 data, whose distribution will stop in mid-2016. Do not use this API for new developments.

Parameter	Description	Possible Value
`SORT`	The *Sort* attribute determines how the elements will be sorted in the returned array.	`SORT:ASC`: returns event data in ascending (chronological) order `SORT:DES`: returns event data in descending (reverse-chronological) order (default)
`START`	The *Start Date* parameter fixes the first date for which historical data should be retrieved.	`START:ddmmmyy`: sets a start date of ddmmmyy
`ZERO`	The *Zero* attribute dictates how entries containing zero values should be handled.	`ZERO:NA:` provide the value 0 (zero) for zero values (default) `ZERO:REPEAT:` copy earlier valid non-zero to this entry, if any `ZERO:SKIP:` drop the entry in the returned array

Arguments

None

AdxRtHistory RunStatus

This property allows the application to know whether the object is currently retrieving historical data.

Arguments

None

Return value

This function returns one of the values listed in the RT_RunStatus enumeration.

AdxRtHistory Source

The name of the data source as a `String`. It must be set in order to enable time series data requests.

Arguments

None

Important! `AdxRtHistory` relies on TS1 data, whose distribution will stop in mid-2016. Do not use this API for new developments.

AdxRtHistory Methods

AdxRtHistory FlushData

This method causes the history object to forget all data retrieved through a previous call to AdxRtHistory RequestHistory.

Arguments

None

AdxRtHistory RequestHistory

This method requests time series data for the item specified by the AdxRtHistory ItemName property. Historical data is retrieved for the fields listed in the parameter FieldList (the actual list of fields retrieved will be stored in the AdxRtHistory ExistingFields property). This action occurs asynchronously; the arrival of history data, or an indication of a request error, causes the AdxRtHistory OnUpdate event to occur.

Arguments

`FieldList` An array of field names as Strings, identifying the fields for which historical values are to be retrieved. If empty, all fields available in the time series are retrieved.

AdxRtHistory Events

AdxRtHistory OnUpdate

This callback handles updates to the requested historical data following a call to the AdxRtHistory RequestHistory method. It may be called a number of times as updates occur, filling the AdxRtHistory Data property with historical event information before the retrieved time series is complete (according to the constraints applied using the AdxRtHistory Mode property).

Arguments

`DataStatus` The status of the data retrieved by the request. This is a value taken from the "RT_DataStatus" on page 105 enumeration.

AdxRtSourceList

AdxRtSourceList Properties

AdxRtSourceList Active

This property retrieves the connection status. The value returned is `True` if the connection is initialized.

Arguments

None

VBA sample

```
Dim res As Boolean
Dim SrcList As AdfinXRtLib.AdxRtSourceList
Set SrcList = createAdxRtSourceList
res = SrcList.Active
```

AdxRtSourceList FieldDefinition

This property retrieves the array of field properties (`Name`, `LongName`, `FID`, `Type`, and `Length`) based on the field name list and FID list. Wildcards can be used to retrieve information for all the fields. Field names and IDs can be mixed in the array.

Arguments

`Fields` A list of one or more field names or IDs as a Variant, identifying the fields for which additional information must be retrieved.

VBA sample

```
Dim count As Integer
Dim res As Variant
Dim MinBound As Integer
```

```
Dim MaxBound As Integer
Dim FieldType As String
Dim OtherValues As String
Dim SrcList As AdfinXRtLib.AdxRtSourceList
Set SrcList = NewAdfinXRtLib.AdxRtSourceList
res = SrcList.FieldDefinition("IDN", "*")
If VarType(res) <> vbEmpty Then
    MinBound = LBound(res)
    MaxBound = UBound(res)
    For i = MinBound To MaxBound
     For j = 0 To 4
       If j = 3 Then 'handle field type
         FieldType = Choose(res(i, j), "RT_TYPE_NUM", "RT_TYPE_STRING", _
           "RT_TYPE_INTEGER", "RT_TYPE_DATE", "RT_TYPE_TIME", _
           "RT_TYPE_TIMSECS", "RT_TYPE_ET")
         Else ' handle field ID , name, long name, length
            OtherValues = res(i, j)
         End If
    Next j
  Next i
End If
```

AdxRtSourceList SourceCount

The property retrieves the number of available sources.

Arguments

None

VBA sample

```
Dim res as integer
Dim SrcList As AdfinXRtLib.AdxRtSourceList
Set SrcList = createAdxRtSourceList
res = SrcList.SourceCount
```

AdxRtSourceList SourceList

This property retrieves the array of available sources.

Arguments

None

VBA sample

```
Dim res As Variant
Dim count As Integer
Dim sourcename As String
Dim SrcList As AdfinXRtLib.AdxRtSourceList
Set SrcList = createAdxRtSourceList

count = SrcList.SourceCount
res = SrcList.SourceList
For i = 0 To count - 1
 sourcename = res(i)
Next i
```

AdxRtSourceList SourceStatus

This property retrieves the status of the specified source.

Arguments

Source	The name of the source.

VBA sample

```
Dim Value as String
Dim status As Integer
Dim SrcList As AdfinXRtLib.AdxRtSourceList
Set SrcList = createAdxRtSourceList

status = SrcList.SourceStatus("IDN")
Value = Choose(status+1, "UP", "DOWN", "INVALID", "UNDEFINED", "NOT SET")
```

AdxRtSourceList Methods

There are no methods for AdxRtSourceList.

AdxRtSourceList Events

AdxRtSourceList OnStatusChange

This is called when the status of a source changes.

Arguments

Name	The name of the source
SourceStatus	The status of the source

Text	Eventual error message linked to the failure status

VBA sample

```
Dim Withevents SrcList as AdfinXRtLib.AdxRtSourceList

Private Sub CommandButton1_Click()
Set SrcList = createAdxRtSourceList
End Sub

Private Sub SrcList_OnStatusChange(ByVal Name As String, _
  ByVal SourceStatus As AdfinXRtLib.RT_SourceStatus, ByVal text As String)
        dim res as string
        MsgBox Choose(SourceStatus+1, "UP", "DOWN", "INVALID",
"UNDEFINED", _
               "NOT SET") & " : " & text, , Name
End Sub
```

AdxRtxLib Parameters and Constants

RT_ItemStatus

This enumeration indicates the status of an individual item retrieved from the source.

Values

`RT_ITEM_OK`	The item is valid.
`RT_ITEM_INVALID`	The item is not available from the source or is not registered.
`RT_ITEM_UNKNOWN`	The item is temporarily in an unknown state.
`RT_ITEM_STALE`	The item is known by the source but is not available, data associated with it may be out of date.
`RT_ITEM_DELAYED`	The item is not permissioned, data associated with it may be out of date.
`RT_ITEM_NOT_PERMISSIONED`	The item is not permissioned.

RT_FieldStatus

This enumeration indicates the status of a field in an item retrieved from the source. Fields must be defined in the data dictionary file (`global.mfl`), otherwise they will not be recognized.

Values

RT_FIELD_OK	The field is present in the field list of its item.
RT_FIELD_INVALID	The field is not present in the item.
RT_FIELD_UNKNOWN	The field is temporarily in an unknown state.
RT_FIELD_UNDEFINED	The field is not defined in the field dictionary of the source.

RT_SourceStatus

This enumeration indicates the status of the source used to retrieve the requested information.

Values

RT_SOURCE_UP	The data source is available.
RT_SOURCE_DOWN	The data source is unavailable.
RT_SOURCE_INVALID	The data source is invalid on this system.
RT_SOURCE_UNDEFINED	The data source is not yet defined.
RT_SOURCE_NOT_SET	The Source property of the object is not assigned.

RT_ListStatus

This enumeration indicates the status of the object itself.

Values

RT_LIST_INACTIVE	No real-time data items are being requested at present.
RT_LIST_RUNNING	Real-time data items are receiving data images or updates.
RT_LIST_UPDATES_STOPPED	Update events for real-time data items have been temporarily stopped.
RT_LIST_LINKS_CLOSED	Updates for the real-time data items are no longer being supplied by the source.

RT_ItemRowView

This enumeration indicates how items in the list should be selected.

Values

RT_IRV_UPDATED	The item array is to contain rows only for those items in the list that are updated.
RT_IRV_ALL	The item array is to contain rows for all items in the list.

RT_FieldRowView

This enumeration indicates how fields in an item should be selected.

Values

RT_FRV_EXISTING:	The field array is to contain rows for all fields in the item.
RT_FRV_UPDATED:	The field array is to contain rows only for those fields in the item that were updated.
RT_FRV_ALL:	The field array is to contain rows for all fields registered in the list for the item.

RT_ItemColumnView

This enumeration indicates which information relating to the items in the list should be selected.

Values

RT_ICV_STATUS:	The item array is to contain a column holding the status of each item as a value from the "RT_ItemStatus" on page 102 enumeration.
RT_ICV_USERTAG:	The item array is to contain a column for the item user tags set using the "AdxRtList UserTag" on page 67 property.

RT_FieldColumnView

This enumeration indicates which information relating to the fields in the item should be selected.

Values

RT_FCV_USERTAG	The field array is to contain a column for the field user tags set using the AdxRtList UserTag property.
RT_FCV_STATUS	The field array is to contain a column holding the status of each field as a value from the RT_FieldStatus enumeration.
RT_FCV_VALUE	The field array is to contain a column holding the value of each field as a variant.

RT_RunMode

This enumeration indicates how update signaling should be performed. They are also passed to the AdxRtList OnStatusChange event callback for information.

Values

RT_MODE_ONTIME_IF_UPDATED	At regular intervals, defined by the "FRQ" attribute of the AdxRtList Mode property, all registered items are to be checked to see if any is updated by the data source since the previous check. For each updated item an AdxRtList OnUpdate event is to be generated.
RT_MODE_ONTIME	AdxRtList OnTime events are to be generated at regular intervals, defined by the "FRQ" attribute of the AdxRtList Mode property, whether or not any data updates are received for the items in the list.
RT_MODE_ONUPDATE	AdxRtList OnUpdate events are to be generated for each item in the list every time data updates are received for that item from the source.

RT_MODE_IMAGE	An AdxRtList OnImage event is to be generated once a single data image is received for each of the items in the list, or after a time-out.
RT_MODE_NOT_SET	No update run mode is set using the AdxRtList StartUpdates method.

See also

"AdxRtList CloseAllLinks" on page 69
"AdxRtList StartUpdates" on page 72
"AdxRtList StopUpdates" on page 73

RT_DataStatus

This enumeration lists the different values used to describe the status of data retrieved by the objects.

RT_DS_FULL	This value indicates that all desired data have is retrieved and is available.
RT_DS_PARTIAL	This value indicates that some of the requested data is available, but that the request has not yet been entirely satisfied.
RT_DS_NULL_ERROR	This value indicates that an error has occurred and that, consequently, no data is available.
RT_DS_NULL_EMPTY	This value indicates that the request returned no data.
RT_DS_NULL_TIMEOUT	This value indicates that a timeout occurred before any data could be retrieved.

RT_RunStatus

This enumeration lists the different states of an object regarding its requests to the source.

RT_RS_READY	The object is ready to be used to request data from a data source.
RT_RS_BUSY	The object is waiting for a response from the server. For objects that accept multiple data updates from the source, this means that more update events can be expected.
RT_RS_NOT_INIT	The object does not have enough information to be able to make a request.

RT_DebugLevel

This enumeration lists the valid values for the AdxRtList DebugLevel property.

Values

RT_DEBUG_NO	No debug messages are to be sent. Exceptions indicate invalid parameters and object event interfaces are used to signal system errors.
RT_DEBUG_IMMEDIATE	The validity of all parameters is checked before the client application actions are allowed to proceed. This may require waiting for responses from remote data sources. Debug messages are to be generated immediately following the actions that provoke them.

AdxErrorMode

This enumeration lists the valid values for the `ErrorMode` property in each of the `AdxRtxLib` objects.

Values

`NO_EXCEPTION`	No action is taken when an error is detected. In this mode, the client application must consult the value of the `ErrorCode` property to know if an error has been encountered.
`DIALOGBOX`	A dialog box is displayed when an error occurs.
`EXCEPTION`	An exception is thrown when an error occurs.

AdxAttrRtList

This enumeration provides identifiers for the attributes of AdxRtList object. These are changed indirectly by supplying a new string to the AdxRtList Mode property of the object.

Values

`ATTR1L_RTLIST_FRQ`	This value identifies the frequency of regular AdxRtList OnTime events. In the Mode string, this is set using the "FRQ" keyword.
`ATTR1L_RTLIST_TIMEOUT`	This value identifies the length of the timeout period after which requests for item data images are to be abandoned. In the `Mode` string, this is set using the `"TIMEOUT"` keyword.
`ATTR1L_RTLIST_SEND_EXTENDED_ITEM_STATUSES`	This value identifies the permission status and determines if this value is to be sent to the client. In the Mode string, this is set using the "`SEIS`" keyword.

AdxAttrRtContribute

This enumeration provides identifiers for the attributes of the AdxRtContribute object. These are changed indirectly by supplying a new string to the AdxRtContribute Mode property of the object.

Values

`ATTR1E_RTCTRB_SCOPE`	This value determines whether the contribution should be local (updating a value in the local cache) or should be sent to the server. The keyword "`SCOPE`" is not used by ActiveX.

AdxAttrRtChain

This enumeration provides identifiers for the attributes of the "AdxRtChain" on page 86 object. These are changed indirectly by supplying a new string to the "AdxRtChain Mode" on page 88 property of the object.

Values

Value	Identifies	Equivalent Mode keyword
ATTR1E_RTCHAIN_IGNE	This value identifies the *Ignore Empty* attribute.	In the Mode string, this is set using the "IGNE" keyword.
ATTR1E_RTCHAIN_LAY	This value identifies the *Layout* mode.	This is set using the "LAY" keyword.
ATTR1E_RTCHAIN_LIVE	This value identifies the *Live* attribute.	This is set using the "LIVE" keyword.
ATTR1S_RTCHAIN_RET	This value identifies the *Return* attribute.	This is set using the "RET" keyword.
ATTR1S_RTCHAIN_SKIP	This value identifies the *Skip* attribute. In the Mode string.	In the Mode string, this is set using the "SKIP" keyword.
ATTR1E_RTCHAIN_UWC	This value identifies the *Update When Completed* attribute.	In the Mode string, this is set using the "UWC" keyword.
ATTR1E_RTCHAIN_SEND_EXTENDED_ITEM_STATUSES	This value identifies the permission status and determines if this value is to be sent to the client.	In the Mode string, this is set using the "SEIS" keyword.

AdxAttrRtHistory

This enumeration provides identifiers for the attributes of the AdxRtHistory object. These are changed indirectly by supplying a new string to the AdxRtHistory Mode property of the object.

Values

Value	Identifies	Equivalent Mode keyword
ATTR1E_RTHIST_HEADER	This value identifies the *Header* attribute.	In the Mode string, this is set using the "HEADER" keyword.
ATTR1E_RTHIST_LAY	This value identifies the *Layout* attribute.	In the Mode string, this is set using the "LAY" keyword.
ATTR1E_RTHIST_NULL	This value identifies the *Null* attribute.	In the Mode string, this is set using the "NULL" keyword.
ATTR1S_RTHIST_RET	This value identifies the *Return* parameter.	In the Mode string, this is set using the "RET" keyword.
ATTR1E_RTHIST_ZERO	This value identifies the *Zero* attribute.	In the Mode string, this is set using the "ZERO" keyword.
ATTR1E_RTHIST_SORT	This value identifies the *Sort* attribute.	In the Mode string, this is set using the "SORT" keyword.
ATTR1D_RTHIST_END	This value identifies the *End Date* parameter.	In the Mode string, this is set using the "END" keyword.

Value	Identifies	Equivalent Mode keyword
`ATTR1E_RTHIST_FRQ`	This value identifies the *Frequency* parameter.	In the `Mode` string, this is set using the `"FRQ"` keyword.
`ATTR1I_RTHIST_NBEVENTS`	This value identifies the *Number of Events* parameter.	In the `Mode` string, this is set using the `"NBEVENTS"` keyword.
`ATTR1D_RTHIST_START`	This value identifies the *Start Date* parameter.	In the `Mode` string, this is set using the `"START"` keyword.

RT_FieldType

This enumeration provides identifiers for the field type returned by the AdxRtSourceList FieldDefinition method. See AdxRtSourceList SourceList.

Values

`RT_TYPE _NUM`	Number/Price field
`RT_TYPE _STRING`	Alphanumeric field
`RT_TYPE _INTEGER`	Integer field
`RT_TYPE _DATE`	Date field (dd mm yyyy)
`RT_TYPE _TIME`	Time field (hh:mm)
`RT_TYPE _TIMSECS`	Time field including seconds (hh:mm:ss)
`RT_TYPE _ET`	Enumeration type

DEX 2 Library (Data Engine Library)

DEX2 Overview

DEX2 and its Components

About DEX2

The Data Engine ActiveX Component 2 (DEX2) library is an in-process COM library. This library allows Thomson Reuters Eikon and Thomson Reuters Eikon Excel to access fundamental data and metadata stored in the Thomson Reuters Platform Snapshot Server (SnS). You can use this library only within the context of Thomson Reuters Eikon. It cannot be used under third party or standalone applications.

There are four components exposed by DEX2:

- Dex2Mgr
- MetaDataMgr
- RData
- RDataMgr

You can download a spreadsheet file containing DEX2 code examples from this link.

Dex2Mgr and object lifetime management

The Dex2Mgr component manages the lifetime of some DEX2components. Dex2Mgr provides methods that must be called so that the application codes retrieve data successfully.

Prior to making any request to Snapshot Server (SnS), user application codes first need to acquire a reference to Dex2Mgr and then initialize its internal configuration and caching system using the Initialize method. Dex2Mgr isnow ready and can be used to create an RData object to request the required fundamental data from SnS.

When the RData object is no longer needed, the user application codes must call the Finalize method of Dex2Mgr to release resources held by the object and to perform all the necessary clean-up operations.

Usage of MetaDataMgr

MetaDataMgr is designed for internal use by Thomson Reuters Eikon and Thomson Reuters Eikon Excel.

Important!	Thomson Reuters recommends that user application codes not use this object.

RData as a data retrieval object

The RData component allows user application codes to request fundamental data from SnS. It provides a mechanism to specify and retrieve data of the instruments of interest. You can configure RData to provide error information if data retrieval or any other failure occurs.

The data retrieved from SnS is stored in the property of the object. This property is called data of type variant and contains a multi-dimensional array of variant values. The number of dimensions of the array depends on the requests for:

- number of instruments
- number of fields
- specified parameters

RDataMgr and the caching mechanism

DEX2 has an internal caching mechanism to accelerate data retrieval,which is purged when it is refreshed. You can configure this using Setting Dialog inside Thomson Reuters Eikon Excel. The internal cache has limited functionality. The user application codes can manually refresh by calling the RefreshCache function of RDataMgr.

Working with DEX2

Using DEX2

In-Memory COM activation

DEX2 is a COM library, but you do not need to register it with the system registry. This is because a registration-free COM activation context provides the binding and activation information that the COM runtime needs as an in-memory structure. Thomson Reuters Eikon creates this memory structure.

How to add a reference to DEX2

You must add a reference to DEX2 to make it available in user application codes.

1 Open the *Visual Basic Editor*.

2 Choose *Tools* > *References* > *Browse*.

3 Locate the `Dex2.dll` file under the `Program` folder of Thomson Reuters Eikon.

> By default, the folder is `C:\Program Files (x86)\Thomson Reuters\Eikon\X\Bin`

DEX2 availability outside the application

DEX2 cannot be used by a standalone Visual Basic or Visual C++ application outside Thomson Reuters Eikon.

Using DEX2 in local mode

There are some limitations to using DEX2 APIs when Thomson Reuters Eikon Excel is operating in local mode. See "Working in Local Mode" on page 120.

Instantiating DEX2 Components

Creating Dex2Mgr

Use the CreateReutersObject function in `PLVbaApis.dll` to create a Dex2Mgr object instance.

> You cannot use the VBA `New` keyword to create an instance of Dex2Mgr.

The sample code shows the Dex2Mgr instantiation:

```
' Put this code in a standard VBA module.
Public Declare _
    Function CreateReutersObject _
    Lib "PLVbaApis.dll" (ByVal progID As String) As Object

Public Function CreateDex2Manager() As DEX2.Dex2Mgr
    Set CreateDex2Manager = CreateReutersObject("Dex2.Dex2Mgr")
End Function

' Declare a variable and instantiate an object
' using the function defined in the previous step
Private m_dex2Mgr As DEX2.Dex2Mgr

Private Sub Main()
    m_dex2Mgr = CreateDex2Manager()
End Sub
```

Creating Dex2Mgr

In Thomson Reuters Eikon Desktop, you can instantiate Dex2Mgr using the `new` keyword.

```
' Declare a variable and instantiate an object
Private m_dex2Mgr As DEX2.Dex2Mgr
Private Sub Main()
 Set m_dex2Mgr = new DEX2.Dex2Mgr
End Sub
```

However, inThomson Reuters Eikon Excel, you cannot instantiate Dex2Mgr using the `new` keyword. You must use the `CreateReutersObject` function in `PLVbaApis.dll` to create a Dex2Mgr object instance.

```
' Put this code in a standard VBA module.
Public Declare _
   Function CreateReutersObject _
   Lib "PLVbaApis.dll" (ByVal progID As String) As Object
Public Function CreateDex2Manager() As DEX2.Dex2Mgr
    Set CreateDex2Manager = CreateReutersObject("Dex2.Dex2Mgr")
End Function
' Declare a variable and instantiate an object
    Private m_dex2Mgr As DEX2.Dex2Mgr
Private Sub Main() m_dex2Mgr = CreateDex2Manager()
End Sub
```

Creating RData

Use the CreateRData function provided by Dex2Mgr to create an RData object instance. However, this function must be called after the call to the Initialize function of Dex2Mgr is completed. The sample code shows the RData instantiation:

```
' Declaring variables
Private m_dex2Mgr As DEX2.Dex2Mgr
Private m_cookie As Long
Private WithEvents m_rdata As DEX2.RData
' Assume that we put this code in a VBA class module
Private Sub Class_Initialize()
   m_dex2Mgr = CreateDex2Manager()
   ' Must call Initialize() at once.
   m_cookie = m_dex2Mgr.Initialize()
   ' We can then create an RData object
   m_rdata = m_dex2Mgr.CreateRData(m_cookie)
   ' Start using the object...
    ' ...
   ' ...
End Sub
```

Creating RDataMgr

Use the CreateRDataMgr function provided by Dex2Mgr to create an RDataMgr object. However, this function must be called after the call to the `Initialize` function of Dex2Mgr is completed. The sample code shows the CreateRDataMgr instantiation:

```
' Declaring variables
Private m_dex2Mgr As DEX2.Dex2Mgr
Private m_cookie As Long
Private m_rdataMgr As DEX2.RDataMgr
 ' Assume that we put this code in a VBA class module
Private Sub Class_Initialize()
   m_dex2Mgr = CreateDex2Manager()
    ' Must call Initialize() at once.
   m_cookie = m_dex2Mgr.Initialize()
    ' We can then create an RDataMgr object
   m_rdataMgr = m_dex2Mgr.CreateRDataMgr(m_cookie)
    ' Start using the object...
    ' ...
   ' ...
End Sub
```

Using the RData Object to Get Data

Set key properties of RData

Before you use the RData object, set its properties including:

- InstrumentIDList
- FieldList

The values assigned to these two properties identify the instrument(s) and fundamental data of interest. Snapshot Server (SnS) uses fields to represent different types of fundamental data.

The application code can request a single instrument with a single field or multiple instruments with multiple fields at the same time. The code shows how values might be assigned to the two properties:

```
rdata = m_dex2Mgr.CreateRData(m_cookie)
 ' Assigning a single instrument and field
rdata.InstrumentIDList = "TRI.N"
rdata.FieldList = "RI.ID.RIC"
' It's also possible to assign multiple instruments or fields
rdata.InstrumentIDList = "TRI.N; GOOG.O; MSFT.O"
rdata.FieldList = "RI.ID.RIC; RI.ID.WERT"
```

Set optional properties of RData

If required, you can also set two optional properties:

- RequestParam
- DisplayParam

Most of the fundamental data requires additional parameterized information, which is a string value assigned to RData using the RequestParam property. For example, to retrieve a historic 5-year average P/E ratio of an instrument, the RequestParam could be set to specify the Financial Period the user wishes to retrieve. It could also be set to specify the Sampling Frequency identifying the number of data (P/E ratio) to be retrieved. The VBA code for assigning these request parameters looks like:

```
rdata = m_dex2Mgr.CreateRData(m_cookie)
' We are interested in the 5-year average P/E ratio of
' Thomson Reuters ordinary share
rdata.InstrumentIDList = "TRI.N"
rdata.FieldList = "RFA.VAL.PRICE2EPS_AAVG5"
' Request parameters:
' (1) Financial Period: Fiscal year 2008 - 2009
' (2) Sampling Frequency: Weekly
rdata.RequestParam = "FP:FY2008;FY2009 SF:W"
```

If there is no request parameter specified, SnS will use its own default request parameters to respond.

Also note that request parameters are field-specific; not every parameter supports every fundamental data. SnS provides the metadata that describes the relation between fundamental data and its corresponding parameters. However, explaining the content of the metadata is beyond the scope of this document.

Set parameters using Insert Function Wizard

An alternative to specify request parameters is to use Thomson Reuters Eikon Excel - Insert Function Wizard.

Application codes can set the DisplayParam property to specify how SnS should format the response data in its content, sorting order, and the layout of the returned table. The code shows how to set the DisplayParam property so that the response from SnS contains the:

- requested instrument name(s) as the row header
- requested field name(s) as the column header
- resulting table transposed

```
rdata = m_dex2Mgr.CreateRData(m_cookie)
 ' We are interested in general fundamental information
' for Thomson Reuters and Microsoft
rdata.InstrumentIDList = "TRI.N; MSFT.O"
' Look for:
' (1) Company name
' (2) Number of employees
' (3) Contact email address
rdata.FieldList = _ "RF.G.COMPNAME; RF.G.NUMEMPLOY; RF.G.CNTEMAIL"
' Here we specify how the reponse table looks like
rdata.DisplayParam = "RH:In CH:Fd Transpose:Y"
```

Display parameters, such as request parameters, are specific to the fundamental field. Use *Thomson Reuters Eikon Excel - Insert Function Wizard* to find the parameters applicable to different fundamental data.

It is possible to assign values to all the properties mentioned using the `SetParameter` function:

```
rdata = m_dex2Mgr.CreateRData(m_cookie)
' Assigning all properties with a single function
rdata.SetParameter _
   "TRI.N; MSFT.O", _
   "RF.G.COMPNAME; RF.G.NUMEMPLOY; RF.G.CNTEMAIL", _
   "", _
   "RH:In CH:Fd Transpose:Y"
```

VBA sample

A complete example is given to show how to use DEX2 objects to request fundamental data and display the result:

```
' Microsoft Excel Objects (Sheet1)
Private m_dex2 As CDex2

Private Sub cmdCancelRequest_Click()
    m_dex2.CancelRequest()
End Sub

Private Sub cmdCreateDex2_Click()
    m_dex2 = New CDex2
End Sub

Private Sub cmdDeleteDex2_Click()
    m_dex2 = Nothing
End Sub

Private Sub cmdRequest_Click()
    m_dex2.Request()
    m_dex2.WaitforResponse()

    If m_dex2.Status = Dex2Lib.DE_DS_FULL Then
        DisplayData(m_dex2.Data)
    End If
End Sub

Private Sub cmdSetErrorHandling_Click()
    ' We can choose to display error code
    m_dex2.SetErrorHandling DE_EH_ERROR_CODES

    ' Or display error description
    ' m_dex2.SetErrorHandling DE_EH_STRING
End Sub
Private Sub DisplayData(ByVal a_data As Object)
    Dim i As Long
    Dim j As Long
    For i = LBound(a_data, 1) To UBound(a_data, 1)
        Dim l_row As Integer
        l_row = i - LBound(a_data, 1) + 1
        For j = LBound(a_data, 2) To UBound(a_data, 2)
            Dim l_col As Integer
            l_col = j - LBound(a_data, 2) + 1

            ' Check whether there are any in the response
            If VarType(a_data(i, j)) = vbError Then
                ActiveCell.Worksheet.Cells(l_row, l_col).Value = _
                                        "ERR MSG: " &
m_dex2.GetErrorString(CLng(a_data(i, j)))
            Else
                ActiveCell.Worksheet.Cells(l_row, l_col).Value = _
                                        a_data(i, j)
            End If
        Next j
    Next i
End Sub
```

```
Private Sub DisplayData(ByVal a_data As Object)
    Dim i As Long
    Dim j As Long
    For i = LBound(a_data, 1) To UBound(a_data, 1)
        Dim l_row As Integer
        l_row = i - LBound(a_data, 1) + 1
        For j = LBound(a_data, 2) To UBound(a_data, 2)
            Dim l_col As Integer
            l_col = j - LBound(a_data, 2) + 1

            ' Check whether there are any in the response
            If VarType(a_data(i, j)) = vbError Then
                ActiveCell.Worksheet.Cells(l_row, l_col).Value = _
                                            "ERR MSG: " &
m_dex2.GetErrorString(CLng(a_data(i, j)))
            Else
                ActiveCell.Worksheet.Cells(l_row, l_col).Value = _
                                            a_data(i, j)
            End If
        Next j
    Next i
End Sub

' Modules (PLVbaApis)

Public Declare Function CreateReutersObject Lib "PLVbaApis.dll" _
    (ByVal progID As String) As Object

Public Function CreateDex2Manager() As Dex2Lib.Dex2Mgr
    CreateDex2Manager = CreateReutersObject("Dex2.Dex2Mgr")
End Function
' Class Modules (CDex2)

Private m_dex2Mgr As Dex2Mgr
Private WithEvents m_rdata As RData
Private m_rdataMgr As RDataMgr
Private m_logger As CLogger
Private m_cookie As Long
Private m_hasReceivedResponse As Boolean
Private m_dataStatus As DEX2_DataStatus
Private m_error As Object

Private Sub Class_Initialize()
    m_dataStatus = DE_DS_NULL_EMPTY
    m_hasReceivedResponse = False

    m_logger = New CLogger
    m_dex2Mgr = CreateDex2Manager()
    m_cookie = m_dex2Mgr.Initialize(m_logger)

    m_rdata = m_dex2Mgr.CreateRData(m_cookie)
    m_rdataMgr = m_dex2Mgr.CreateRDataMgr(m_cookie)
End Sub

Public Sub SetErrorHandling(ByVal a_errorHandling As DEX2_ErrorHandling)
    m_dex2Mgr.SetErrorHandling(m_cookie, a_errorHandling)
End Sub

Public Function GetErrorString(ByVal a_errorCode As Long)
    GetErrorString = m_dex2Mgr.GetErrorString(a_errorCode)
End Function
Private Sub Class_Terminate()
    m_dex2Mgr.Finalize(m_cookie)
    m_rdata = Nothing
    m_dex2Mgr = Nothing
End Sub
```

```
Public Function Request()

    On Error GoTo ErrorHandler

    m_hasReceivedResponse = False

    ' We can set all required values using the following properties
    m_rdata.InstrumentIDList = "TRI.N; MSFT.O, GOOG.O"
    m_rdata.FieldList = "RF.IS.NetSales"
    m_rdata.RequestParam = "FP:FY2010;FY2005 CURR:EUR"
    m_rdata.DisplayParam = "RH:In CH:Fd"

    ' Or use just only this function
    ' m_rdata.SetParameter _
    '     "TRI.N, MSFT.O, GOOG.O", _
    '     "RF.IS.NetSales", _
    '     "FP:FY2010;FY2005 CURR:EUR", _
    '     "RH:In CH:Fd"

    ' Ignore cache; get data directly from the Snapshot Server
    m_rdata.Subscribe(False)

    ' Or use cache by default
    ' m_rdata.Subscribe
    Exit Function
ErrorHandler:
    MsgBox(m_dex2Mgr.GetErrorString(Err.Number))
End Function

Public Function WaitforResponse()
    Do While Not m_hasReceivedResponse
        DoEvents()
    Loop
End Function

Public Function CancelRequest()
    If (m_rdata.RunStatus = DE_RS_BUSY) Then
        m_rdata.CancelRequest()
    End If
End Function

Public Property Get Data() As Variant
    Data = m_rdata.Data
End Property

Public Property Get Status() As DEX2_DataStatus
    Status = m_dataStatus
End Property

Public Property Get Error() As Variant
    Error = m_error
End Property

Private Sub m_rdata_OnUpdate(ByVal a_dataStatus As Dex2Lib.DEX2_DataStatus, ByVal
a_error As Object)
    ' Indicates that data has been received
    m_hasReceivedResponse = True

    m_dataStatus = a_dataStatus
    m_error = a_error
End Sub
' Class Modules (CLogger)

Implements Dex2Lib.IDex2Logger

Private Sub IDex2Logger_LogMessage( _
        ByVal dex2lsLogSeverity As Dex2Lib.DEX2_LogSeverity, _
        ByVal bstrLogMessage As String)

    MsgBox("Severity = " & dex2lsLogSeverity & _
        " Error description = " & bstrLogMessage)

End Sub
```

Working in Local Mode

Local mode is the offline mode of Thomson Reuters Eikon Excel where the application runs without connecting to the Thomson Reuters platform. In local mode, you can continue to view real-time data using locally deployed feeds.

Activation of local mode

If you are unable to connect to the Thomson Reuters platform, you can sign in to Thomson Reuters Eikon Excel in local mode. If you are already connected to the Thomson Reuters platform and you get disconnected, Thomson Reuters Eikon Excel automatically switches to local mode.

DEX2 API usage in local mode

In local mode, DEX2 objects can be instantiated normally. However, if any of their methods are called, Thomson Reuters Eikon Excel displays an error. The methods can be used once you reconnect to the Thomson Reuters platform.

DEX2Mgr Object

About Dex2Mgr

Overview

Dex2Mgr manages the lifetime of RData and RDataMgr components. It provides functions to create RData and RDataMgr, and converts a numeric error code to its corresponding string description. It also allows the user application code to specify the format of an error; usually as a numeric value or string.

CreateRData

Call Create RData object(lCookie) or Subscribe to SnapshotServer

Use this function to create an RData object to subscribe to Snapshot Server (SnS) for data. To do so, retrieve the cookie of type `Long`, from the Initialize function, as the input parameter to create an RData object. It is possible to create more than one RData object using the same cookie.

Arguments

Argument	Description
`ICookie`	This input parameter is a value of type `Long` retrieved from the `Initialize()` function called prior to this function.

VBA sample

```
Private m_dex2Mgr As DEX2.Dex2Mgr
Private m_cookie As Long
Private WithEvents m_rdata As DEX2.RData

Private Sub Class_Initialize()
    m_dex2Mgr = CreateDex2Manager()
    m_cookie = m_dex2Mgr.Initialize()
End Sub

Public Function Subscribe( _
    ByVal a_instrument As String, _
    ByVal a_field As String, _
    ByVal a_requestParameters As String, _
    ByVal a_displayParameters As String)

    m_rdata = m_dex2Mgr.CreateRData(m_cookie)

    m_rdata.InstrumentIDList = a_instrument
    m_rdata.FieldList = a_field
    m_rdata.RequestParam = a_requestParameters
    m_rdata.DisplayParam = a_displayParameters

    m_rdata.Subscribe(False)
End Function

Private Sub m_rdata_OnUpdate(ByVal DataStatus As Dex2Lib.DEX2_DataStatus, ByVal
Error As Variant)
    ' Do something here
    ' ...
    ' ...
End Sub
```

CreateRDataMgr

Call CreateRDataMgr(lCookie) or Instantiate RDataMgr

Use this function to create an instance of RDataMgr by retrieving the cookie of type `Long` as the input parameter from the Initialize function.

Arguments

Argument	Description
`ICookie`	This input parameter is a value of type `Long` retrieved from the `Initialize()` function called prior to this function.

VBA sample

```
Private m_dex2Mgr As DEX2.Dex2Mgr
Private m_cookie As Long
Private m_rdataMgr As DEX2.RDataMgr

Private Sub Main()
    m_dex2Mgr = CreateDex2Manager()
    m_cookie = m_dex2Mgr.Initialize()
End Sub

' Use RDataMgr to refresh DEX2 cache
Private Sub RefreshCache()
    m_rdataMgr = m_dex2Mgr.CreateRDataMgr(m_cookie)
    m_rdataMgr.RefreshCache(True)
End Function
```

Finalize

Call Finalize(lCookie) or Invalidate the DEX2 context

Use this functionto invalidate the DEX2 context represented by the `lCookie` input parameter. CreateRData and CreateRDataMgr use the cookie of type `Long` from the Initialize function to create RData and RDataMgr objects. The object context value refers to the working context created by this class. You free the created objects using the same context. It is possible to create more than one object of the same type using the same cookie.

Arguments

Argument	Description
`ICookie`	This input parameter is a value of type `Long` retrieved from the `Initialize()` function called prior to this function.

VBA sample

```
Private m_dex2Mgr As DEX2.Dex2Mgr
Private m_cookie As Long
Private WithEvents m_rdata As DEX2.RData

Private Sub Class_Initialize()
    Set m_dex2Mgr = CreateDex2Manager()
    m_cookie = m_dex2Mgr.Initialize()
    m_rdata = m_dex2Mgr.CreateRData(m_cookie)
End Sub

Private Sub Class_Terminate()
    m_rdata.CancelRequest
    Set m_rdata = Nothing
    m_dex2Mgr.Finalize(m_cookie)
    Set m_dex2Mgr = Nothing
End Function
```

Dex2Mgr GetErrorString

Use GetErrorString(_errorCode) or Obtain error description

Use this function to obtain the description of the error represented by the `a_errorCode` numeric value. All DEX2 functions can return errors. If an error occurs during the data retrieval process, it comes from the OnUpdate callback.

Arguments

Argument	Description
`a_errorCode`	A numeric error code

Return value

The return value is the associated description of the specified error code as a string value.

VBA sample

```
Private m_dex2Mgr As Dex2Lib.Dex2Mgr
Private m_cookie As Long
Private WithEvents m_rdata As Dex2Lib.RData

Private Sub Class_Initialize()
    Set m_dex2Mgr = CreateDex2Manager()
    m_cookie = m_dex2Mgr.Initialize()

    ' Set error format to return error code
    m_dex2Mgr.SetErrorHandling m_cookie, DE_EH_ERROR_CODES
End Sub

Public Sub Request()
    Set m_rdata = m_dex2Mgr.CreateRData(m_cookie)

    ' Specify an invalid field; error code will be given in the callback
    Dim field As String
    field = "RI.ID.RIC444"

    m_rdata.SetParameter "TRI.N", field
End Sub

Private Sub m_rdata_OnUpdate(ByVal DataStatus As Dex2Lib.DEX2_DataStatus,
ByVal
Error As Variant)
    Dim errorcode As Long
    ' Convert Variant to Long for error code
    errorcode = CLng(Error)
    MsgBox "ErrorString = " & m_dex2Mgr.GetErrorString(errorcode)
End Sub
```

Dex2Mgr Initialize

Call Initialize([varlLogger])or Prepare Data Engine

Use this function to prepare the Data Engine to request data from SnSand use the IDex2Logger and its IDex2Logger_LogMessage callback mechanism. The Initialize function returns a numeric value of type `Long`. This value is a cookie that is used to refer to the working context created by this class. The Intialize function can receive an optional `varILogger` parameter, which is the IDex2Logger_LogMessage logger class. When DEX2 generates log information, the severity (same as defined in Deployment Manager of Thomson Reuters Eikon) and its description will be passed into the callback of the logger and can be interpreted by the application. (See Dex2_LogSeverity).

Arguments

Argument	Description
`varlLogger`	An object of any class that implements the IDex2Logger interface.

Return value

Return value	Description
`Long`	The function returns a value of type `Long`. It must be kept for later reference as it is required by the functions: • `CreateRData()` • `CreateRDDataMgr()` • `SetErrorHandling()` • `Finalize()`

Example 1

This example shows when to call the function without the optional `varlLogger` parameter.

```
Private m_dex2Mgr As DEX2.Dex2Mgr
Private m_cookie As Long

Private Sub Class_Initialize()
    m_dex2Mgr = CreateDex2Manager()
    m_cookie = m_dex2Mgr.Initialize()
End Sub
```

Example 2

This example shows when to call the function with the optional `varlLogger` parameter.

```
Implements DEX2.IDex2Logger

Private m_dex2Mgr As DEX2.Dex2Mgr
Private m_cookie As Long

Private Sub Class_Initialize()
    m_dex2Mgr = CreateDex2Manager()
    m_cookie = m_dex2Mgr.Initialize(Me)
End Sub

Private Sub IDex2Logger_LogMessage( _
    ByVal a_severity As DEX2.DEX2_LogSeverity, _
    ByVal a_message As String)
    MsgBox(a_message)
End Sub
```

Dex2Mgr SetErrorHandling

Call SetErrorHandling(lCookie, a_errorHandling) or Set format of DEX2 error

Use this function to specify the format of the DEX2 error that returns inside the Data property of the requested RData. The error formats are:

- as a string (`DE_EH_STRING`)
- as a numeric value (`DE_EH_ERROR_CODES`)

User application code sets the error format prior to sending a request to SnS. DEX2 receives and stores the result of a request in the Data property of the requested RData object. The property has a VBA vbVariant type. When requesting fundamental data for multiple instruments, it is possible that the request fails for some of the instruments. In such a case, the user application code still has a valid Data property (See Data property). However, some elements in the array can contain error information. When the format is set to `DE_EH_STRING`, the array elements that contain error information will be of type `vbString`. When the format is set to `DE_EH_ERROR_CODES`, the array elements that contain error information will be of type `vbError`. The format of the error information follows the setting of the SetErrorHandling method. This method also determines the type of error elements. To convert a numeric error code to the corresponding error description, use the GetErrorString function (see "Dex2Mgr GetErrorString" on page 124). The user application code can use this to determine whether the reponse of the request made earlier contains an error.

Arguments

Argument	Description
`lCookie`	This input parameter is a value of type `Long` retrieved from the `Initialize()` function called prior to this function.
`a_errorHandling`	Possible values of this error format are: • `DE_EH_STRING` (Default) when error is string value • `DE_EH_ERROR_CODES` when errors are numeric values To get the corresponding string description of the error code, use the `GetErrorString` function. See "DEX2_ErrorHandling" on page 143

VBA sample

```
Private m_dex2Mgr As Dex2Lib.Dex2Mgr
Private m_cookie As Long
Private WithEvents m_rdata As Dex2Lib.RData

Private Sub Class_Initialize()
    Set m_dex2Mgr = CreateDex2Manager()
    m_cookie = m_dex2Mgr.Initialize()

    ' Set error format to return error code
    m_dex2Mgr.SetErrorHandling m_cookie, DE_EH_ERROR_CODES
End Sub

Public Sub Request()
    Set m_rdata = m_dex2Mgr.CreateRData(m_cookie)
    ' Specify an invalid field
    m_rdata.SetParameter "TRI.N; GOOG.O", "RI.ID.RIC444"
End Sub

Private Sub m_rdata_OnUpdate(ByVal DataStatus As _
    Dex2Lib.DEX2_DataStatus,ByVal Error As Variant)

    Dim i As Long
    Dim j As Long
    Dim data As Variant

    data = m_rdata.Data

    For i = LBound(data, 1) To UBound(data, 1)
        Dim l_row As Integer
        l_row = i - LBound(data, 1) + 1
        For j = LBound(data, 2) To UBound(data, 2)
            Dim l_col As Integer
            l_col = j - LBound(data, 2) + 1

            ' Check whether there are any in the response
            If VarType(data(i, j)) = vbError Then
                Msgbox "ERR MSG: " & _
                                        m_dex2Mgr.GetErrorString(CLng
(data(i,
j))))
            Else
                Msgbox data(i, j)
            End If
        Next j
    Next i

End Sub
```

RData Object

RData CancelRequest

Member type

Method

Stop request

This allows the client code to stop the request made earlier.

VBA sample

```
Private m_dex2Mgr As Dex2Lib.Dex2Mgr
Private m_cookie As Long
Private WithEvents m_rdata As Dex2Lib.RData
Private m_hasReceivedResponse As Boolean
Private m_counter as Long

Private Sub Class_Initialize()
    Set m_dex2Mgr = CreateDex2Manager()
    m_cookie = m_dex2Mgr.Initialize()
End Sub

Public Sub Request()
    m_hasReceivedResponse = False
    Set m_rdata = m_dex2Mgr.CreateRData(m_cookie)
    m_rdata.SetParameter "TRI.N", "RI.ID.RIC"
    m_rdata.Subscribe

    WaitforResponse

    ' Here we can process the retrieved data
    ' ...
    ' ...
End Sub

Public Function WaitforResponse()
    Do While Not m_hasReceivedResponse
        ' If we wait long enough already, cancel the previuos request
        If ( m_counter > 10000000 ) Then
            m_rdata.CancelRequest
        End If
        ' Otherwise, continue waiting
        DoEvents
    Loop
End Function
Private Sub m_rdata_OnUpdate(ByVal a_dataStatus As  Dex2Lib.DEX2_
DataStatus, ByVal
a_error As Object)

    ' Indicates that data has been received
    m_hasReceivedResponse = True

End Sub
```

RData Data

Member type

Property

Request result as a variant array

This is a read-only variant array containing the result of the request madeearlier to the Snapshot Server (SnS):

`VarType(expression) = vbArray + vbVariant`

This result should be examined together with DEX2_DataStatus (cross reference) received from the OnUpdate event.

When receiving responses from SnS, the Data property can be either:

Response	Description
an invalid object and not to be used	This happens when the request fails for all requested instruments. It is caused when any of these parameters are invalid: the instrument name, field name, request, or display. To determine whether the request is a success and the Data property can be used, verify the value of the DataStatus argument of the OnUpdate callback of the RData object.
a valid object holding a single value or an array of data of type variant	When requesting fundatamental data for multiple instruments, it is possible that the request fails for some of the instruments. In such a case, the user application code still has a valid Data property but some elements in the array will contain error information.

VBA sample

```
Private m_dex2Mgr As Dex2Lib.Dex2Mgr
Private m_cookie As Long
Private WithEvents m_rdata As Dex2Lib.RData

Private Sub Class_Initialize()
    Set m_dex2Mgr = CreateDex2Manager()
    m_cookie = m_dex2Mgr.Initialize()
End Sub

Public Sub Request()
    Set m_rdata = m_dex2Mgr.CreateRData(m_cookie)
    m_rdata.SetParameter "TRI.N; MSFT.O", "RI.ID.RIC"
End Sub

Private Sub m_rdata_OnUpdate(ByVal DataStatus As Dex2Lib.DEX2_DataStatus,
ByVal
Error As Variant)
    Select Case DataStatus
        Case Is = Dex2Lib.DE_DS_FULL
            DisplayData (m_rdata.Data)
        Case Is = Dex2Lib.DE_DS_NULL_ERROR
            MsgBox "DE_DS_NULL_ERROR"
        Case Is = Dex2Lib.DE_DS_NULL_EMPTY
            MsgBox "DE_DS_NULL_EMPTY"
        Case Is = Dex2Lib.DE_DS_NULL_TIMEOUT
            MsgBox "DE_DS_NULL_TIMEOUT"
    End Select
End Sub
```

```
Private Sub DisplayData(a_data As Variant)
    Dim i As Long
    Dim j As Long
    For i = LBound(a_data, 1) To UBound(a_data, 1)
        Dim l_row As Integer
        l_row = i - LBound(a_data, 1) + 1
        For j = LBound(a_data, 2) To UBound(a_data, 2)
            Dim l_col As Integer
            l_col = j - LBound(a_data, 2) + 1

            ActiveCell.Worksheet.Cells(l_row, l_col).Value = a_data(i, j)
        Next j
    Next i
End Sub
```

RData DisplayParam

Member type

Property

Display format properties

Application codes may set DisplayParam to specify how SnS should format the response datato specify the content to be displayed, sorting order, and layout of the returned table.

Display parameters are colon-separated key-value pairs. You can specify multiple display parameters byseparating pairs with *Space*.

VBA sample

Parameter format	Example
A single key-value pair	`KEY:value`
A key with multiple values	`Key:value1;value2`
Multiple key-value pairs	`KEY:value KEY:value`
Multiple key-value pairs; each key with multiple values	`Key:value1;value2 Key:value1;value2`

Row and column headers

SnS MetaData defines the entire list of supported headers or dimensions.

Header	Value	Description
Row	RH	This notifies SnS to insert a row header to the left of the result table. The header is then filled with the value specified in pair with this key.
Column	CH	This notifies SnS to insert a column header on top of the result table. The header is then filled with the value specified in pair with this key.

:

VBA sample

Header	Value	Description
Instrument	In	SnS fills the specified header with the requested instrument(s).
Field	Fd	SnS fills the specified header with the requested field name(s).

Even if headers are not specified, the default layout of data is driven by implicit and invisible positions of In (Instruments) in rows and Fd (Fields) in columns. If only one of these two parameters is specified, then it is In or Fd in the headerswhere the implicit layout of the data places the omitted header iina position opposite to the other.

TRANSPOSE array

The `TRANSPOSE:YES/NO` (`Y/N`) parameter transposes the returned table. For example, for a two-dimensional table with no header row and column, `TRANSPOSE:YES` displays symbols in columns and fields n rows,which is the opposite of the default display.

When specified together with the `RH` or `CH` parameter, the latter takes precedence, where the returned table is first given a row or column header and then transposed.

Possible values:

Item	Value
`NO` or `N`	Do not transpose the returned table (Default).
`YES` or `Y`	Transpose the returned table.

Sorting order of the array

The sorting order applies to all dimensions of the returned table (array) except Instruments and Fields.If `SORT` does not apply to the selected fields, it is ignored (no error message is sent).

Possible values:

Item	Value
`ASC` or `A`	Sort the returned table in ascending order (no default value).
`DESC` or `D`	Sort the returned table in descending order.

Accept the sorting order defined by SnS, which can be different depending on the display parameter.

VBA sample

```
Private m_dex2Mgr As DEX2.Dex2Mgr
Private m_cookie As Long
Private WithEvents m_rdata As DEX2.RData

Private Sub Class_Initialize()
    m_dex2Mgr = CreateDex2Manager()
    m_cookie = m_dex2Mgr.Initialize()
End Sub

Public Sub Request()
    m_rdata = m_dex2Mgr.CreateRData(m_cookie)
    m_rdata.InstrumentIDList = "TRI.N"
    m_rdata.fieldList = "RI.ID.RIC"

    Dim displayParameters As String
    displayParameters = "RH:In CH:Fd"

    m_rdata.DisplayParam = displayParameters
    m_rdata.Subscribe
End Sub
```

RData FieldList

Member type

Property

Information request about one or more fields

SnS refers to different fundamental data as fields. The user application code can request information about a field or multiple fields by assigning a value to FieldList.

As a variant type, this parameter can take a string value with a specific format, an array of strings, or even a variant containing a string array.

To assign multiple fields as a single string value, use comma or semicolon as field separator.

RData uses the value assigned to this property and requests the corresponding fundamental data of the instruments specified by the InstrumentIDList property in SnS.

VBA sample

```
Private WithEvents m_rdata As Dex2Lib.RData

Public Sub Request()
    Set m_rdata = m_dex2Mgr.CreateRData(m_cookie)

    'Sample 1
    Dim fieldList(2) As String
    fieldList(0) = "RI.ID.RIC"
    fieldList(1) = "RE.C.Buy"

    'Sample 2
    Dim fieldList As Variant
    fieldList = Array("RI.ID.RIC", "RE.C.Buy")

    'Sample 3
    Dim fieldList As String
    fieldList = "RI.ID.RIC; RE.C.Buy"

    m_function.InstrumentIDList = "TRI.N"
    m_function.fieldList = fieldList
    m_function.Subscribe

End Sub
```

RData InstrumentIDList

Member type

Property

Set one or more Instruments in an RData request

This sets single or multiple instruments to be used in an RData request. Commas or semi-colons are used as field separators.

VBA sample

```
Private WithEvents m_rdata As Dex2Lib.RData

Private WithEvents m_rdata As Dex2Lib.RData

Public Sub Request()
    Set m_rdata = m_dex2Mgr.CreateRData(m_cookie)

    'Sample 1
    Dim instrumentList(2) As String
    instrumentList (0) = "TRI.N"
    instrumentList (1) = "MSFT.O"

    'Sample 2
    Dim instrumentList As Variant
    instrumentList = Array("TRI.N", "MSFT.O")

    'Sample 3
    Dim instrumentList As String
    instrumentList = "TRI.N; MSFT.O"

    m_function.InstrumentIDList = instrumentList
    m_function.fieldList = "RI.ID.RIC"
    m_function.Subscribe

End Sub
```

RData OnUpdate

Member type

Event

Send status of data update

This callback function informs the user application code after data retrieval is completed for the:

- instrument specified in InstrumentIDList
- field specified in FieldList
- request parameter specified in RequestParam
- display parameter specified in DisplayParam

All retrieved data is now available in the Data property. The number of data columns and rows depends on the request and display parameters.

Arguments

Argument	Description
`DataStatus`	This is a value from the DEX2_DataStatus enumeration identifying the status of data, if any, received in response to a request. See DEX2_DataStatus (cross reference).

Argument	Description
Error	The format of Error can be an error code or error string depending on DEX2_ErrorHandling enumeration in SetErrorHandling.

VBA sample

```
Private m_dex2Mgr As Dex2Lib.Dex2Mgr
Private m_cookie As Long
Private WithEvents m_rdata As Dex2Lib.RData

Private Sub Class_Initialize()
    Set m_dex2Mgr = CreateDex2Manager()
    m_cookie = m_dex2Mgr.Initialize(Me)
End Sub

Public Sub Request()
    Set m_rdata = m_dex2Mgr.CreateRData(m_cookie)
    m_rdata.SetParameter "TRI.N; MSFT.O", "RI.ID.RIC"
    m_rdata.Subscribe
End Sub

Private Sub m_rdata_OnUpdate(ByVal DataStatus As Dex2Lib.DEX2_DataStatus,
ByVal
Error As Variant)
    Select Case DataStatus
        Case Is = Dex2Lib.DE_DS_FULL
            DisplayData (m_rdata.Data)
        Case Is = Dex2Lib.DE_DS_NULL_ERROR
            MsgBox "DE_DS_NULL_ERROR, Error is" & Error
        Case Is = Dex2Lib.DE_DS_NULL_EMPTY
            MsgBox "DE_DS_NULL_EMPTY, Error is " & Error
        Case Is = Dex2Lib.DE_DS_NULL_TIMEOUT
            MsgBox "DE_DS_NULL_TIMEOUT, Error is " & Error
    End Select
End Sub

Private Sub DisplayData(a_data As Variant)
    Dim i As Long
    Dim j As Long
    For i = LBound(a_data, 1) To UBound(a_data, 1)
        Dim l_row As Integer
        l_row = i - LBound(a_data, 1) + 1
        For j = LBound(a_data, 2) To UBound(a_data, 2)
            Dim l_col As Integer
            l_col = j - LBound(a_data, 2) + 1

            ActiveCell.Worksheet.Cells(l_row, l_col).Value = a_data(i, j)
        Next j
    Next i
End Sub
```

RData Refresh

Member type

Method

Request resend

This makes RData resend the request to SnS.

VBA sample

```
Private m_dex2Mgr As DEX2.Dex2Mgr
Private m_cookie As Long
Private WithEvents m_rdata As DEX2.RData

Private Sub Class_Initialize()
    m_dex2Mgr = CreateDex2Manager()
    m_cookie = m_dex2Mgr.Initialize()
End Sub

Public Function Subscribe()
    Dim instrumentList(2) As String
    Dim field As String

    m_rdata = m_dex2Mgr.CreateRData(m_cookie)
    instrumentList(0) = "TRI.N"
    instrumentList(1) = "MSFT.O; GOOG.O"
    field = "RI.ID.RIC"
    m_function.SetParameter instrumentList, field
    m_rdata.Subscribe(False)
End Function

Public Sub Refresh()
    m_rdata.Refresh
End Sub
```

RData RequestParam

Member type

Property

Request parameterized information

Many fundamental data require additional parameterized information, which can be specified with RequestParam. This is an optional parameter.

For example, the RF.IS.NetSales field has the Financial Period (FP) request parameter that allows the user application code to specify the year range in the request. For example, 2005 to 2010 is `FY2010;FY2005`.

In this case, the user application code will get the number of the data row corresponding to the year range. If the Financial Period request parameter is not specified, the user application code will receive only one year, which is a default value returned from SnS.

The format of the RequestParam property string is a space-separated list of keywords with possible associated values. Normally, the request parameter is different for each field. Some fields have multiple request parameters or multiple values associated with a request parameter, while others do not. You can see all request parameters and their values in *Thomson Reuters Eikon Excel - Insert Function Wizard*.

VBA sample

```
Private m_dex2Mgr As DEX2.Dex2Mgr
Private m_cookie As Long
Private WithEvents m_rdata As DEX2.RData

Private Sub Class_Initialize()
    m_dex2Mgr = CreateDex2Manager()
    m_cookie = m_dex2Mgr.Initialize()
End Sub

Public Sub Request()
    m_rdata = m_dex2Mgr.CreateRData(m_cookie)
    m_rdata.InstrumentIDList = "TRI.N"
    m_rdata.fieldList = "RF.IS.NetSales"

    'Sample 1, FP = Financial Period, CURR = Currency
    Dim requestParameters As String
    requestParameters = "FP:FY2010;FY2005 CURR:EUR"

    'Sample 2, FP = Financial Period, CURR = Currency
    Dim requestParameters As Variant
    requestParameters = Array("FP:FY2010;FY2005", "CURR:USD")

    m_rdata.RequestParam = requestParameters
    m_rdata.Subscribe(False)
End Sub
```

RData RunStatus

Member type

Method

View status of RData

This indicates the current status of RData. See DEX2_RunStatus (cross reference).

Return value

Enumerator	Description
DE_RS_NOT_INIT	RData is not in a statethat allows a request to be made. The RData properties should be set accordingly.
DE_RS_READY	RData is in a state that allows a request to be made. This is also the state to which the object will return on completion of a request.

Enumerator	Description
DE_RS_BUSY	RData is currently being used to make a request.

VBA sample

```
RData is currently being used toPrivate m_dex2Mgr As DEX2.Dex2Mgr
Private m_cookie As Long
Private WithEvents m_rdata As DEX2.RData

Private Sub Class_Initialize()
    m_dex2Mgr = CreateDex2Manager()
    m_cookie = m_dex2Mgr.Initialize()
End Sub

Public Function Subscribe()
    m_rdata = m_dex2Mgr.CreateRData(m_cookie)
    'Sample 1
    Dim rdataRunStatus As Dex2Lib.DEX2_RunStatus
    rdataRunStatus = m_rdata.RunStatus

    m_rdata.InstrumentIDList = "TRI.N"
    m_rdata.FieldList = "RI.ID.RIC"
    m_rdata.Subscribe
End Function
make a request
```

RData SetParameter

Member type

Method

Set all parameters simultaneously

Using SetParameter is equivalent to sequentially setting the InstrumentIDList, FieldList, RequestParam, and DisplayParam properties. Use this method for better performance.

Arguments

Argument	Description
InstrumentList	This is a comma- or semi-colon-separated list of Reuters Instrument Codes (RIC) or string array. See "RData InstrumentIDList" on page 134.
FieldList	This is a comma- or semi-colon-separated list of fundamental fields or string array. See "RData FieldList" on page 133.
RequestParam [Optional]	See "RData RequestParam" on page 137.
DisplayParam [Optional]	See "RData DisplayParam" on page 131.

VBA sample

```
Private m_dex2Mgr As DEX2.Dex2Mgr
Private m_cookie As Long
Private WithEvents m_rdata As DEX2.RData

Private Sub Class_Initialize()
    m_dex2Mgr = CreateDex2Manager()
    m_cookie = m_dex2Mgr.Initialize()
End Sub

Public Function Subscribe()
    Dim instrumentList(2) As String
    Dim field As String

    m_rdata = m_dex2Mgr.CreateRData(m_cookie)
    instrumentList(0) = "TRI.N"
    instrumentList(1) = "MSFT.O; GOOG.O"
    field = "RI.ID.RIC"
    m_rdata.SetParameter instrumentList, field
    m_rdata.Subscribe(False)
End Function
```

RData Subscribe

Member type

Method

Create a request

This uses values specified in the InstrumentIDList, FieldList, RequestParam, and DisplayParam properties in order to create a request. By default, RData will look in the cache for data (UseCache = true).

Arguments

Argument	Description
UseCache	• `True` (Default) gets a response from the cache if a similar request was sent before. • `False` makes a new request to SnS.

VBA sample

```
Private m_dex2Mgr As DEX2.Dex2Mgr
Private m_cookie As Long
Private WithEvents m_rdata As DEX2.RData

Private Sub Class_Initialize()
    m_dex2Mgr = CreateDex2Manager()
    m_cookie = m_dex2Mgr.Initialize()
End Sub

Public Function Subscribe(
    ByVal a_instrument As String, _
    ByVal a_field As String, _
    ByVal a_requestParameters As String, _
    ByVal a_displayParameters As String)

    m_rdata = m_dex2Mgr.CreateRData(m_cookie)
    m_rdata.InstrumentIDList = a_instrument
    m_rdata.FieldList = a_field
    m_rdata.RequestParam = a_requestParameters
    m_rdata.DisplayParam = a_displayParameters
    m_rdata.Subscribe(False)
End Function
```

RDataMgr Object

RDataMgr Overview

Overview or shared cache

DEX2 has a single cache in shared memory for all instances of Dex2Mgr inside and outside the process. RDataMgr provides the RefreshCache method to manage this DEX2 internal cache. Use the CreateRDataMgr method of Dex2Mgr to create an RDataMgr object.

Calling this method impacts all other instances of DEX2 you use on any of the Thomson Reuters Eikon products and, therefore, you must use it carefully.

RDataMgr RefreshCache

RefreshCache(ClearAll)or Clear cache

Specifies whether to clear the cache of the associated Dex2Mgr before resending a request of RData objects belonging to the same context.

Arguments

Argument	Description
ClearAll	• `True`: completely clear the cache content and resend requests • `False`: do not clear the cache content before resending requests

VBA sample

```
' Declaring variables
Private m_dex2Mgr As DEX2.Dex2Mgr
Private m_cookie As Long
Private m_rdataMgr As DEX2.RDataMgr

' Assume that we put this code in a VBA class module
Private Sub Class_Initialize()
    m_dex2Mgr = CreateDex2Manager()

    ' Must call Initialize() at once.
    m_cookie = m_dex2Mgr.Initialize()

    ' We can then create an RDataMgr object
    m_rdataMgr = m_dex2Mgr.CreateRDataMgr(m_cookie)

' Call the only method of RDataMgr
    m_rdataMgr.RefreshCache True
End Sub
```

Enumeration

DEX2_DataStatus

Status of RData

The `Dex2_DataStatus` enumeration indicates the status of RData.

Usage

The enumeration used here follows the `OnUpdate` method.

Enumerator	Description
`DE_DS_FULL`	The RData request completes successfully and its Data property contains the requested information. The RunStatus property of the object sets to `DE_RS_READY` once the request completes.
`DE_DS_NULL_ERROR`	The RData request completes unsuccessfully.
`DE_DS_NULL_EMPTY`	The RData request completes successfully but no information is found that corresponds to the request made.
`DE_DS_NULL_TIMEOUT`	The RData request is outstanding for a period of time longer than its timeout period. Due to this, the request is canceled.

DEX2_ErrorHandling

Return error format

The `DEX2_ErrorHandling` enumeration indicates the format of the error.

Usage

This enumeration is used in the `SetErrorHandling` method.

Enumerator	Description
`DE_EH_STRING`	The error format is a string.
`DE_EH_ERROR_CODES`	The error format is a numeric code.

DEX2_RunStatus

Overview

The `Dex2_RunStatus` enumeration indicates the status of RData.

Usage

This enumeration is used in the `RunStatus` method.

Enumerator	Description
`DE_RS_NOT_INIT`	RData is not in a state that allows a request to be made. The RData properties should be set accordingly.
`DE_RS_READY`	RData is in a state that allows a request to be made. This is also the state to which the object will return on completion of a request.
`DE_RS_BUSY`	RData is currently being used to make a request.

DEX2_LogSeverity

Return log severity

The `Dex2_LogSeverity` enumeration indicates log severity provided by DEX2.

Usage

This enumeration is used in the `Initialize` method.

Enumerator	Description
`DE_LS_INFO`	Only information with no impact on the application.
`DE_LS_WARNING`	A warning is logged: the application is still running.
`DE_LS_ERROR`	An error occurs: the application may not continue running.
`DE_LS_DEBUG`	No impact on the application: its purpose is debugging.

Using RSearch COM API

About RSearch COM API

Overview

RSearch COM API provides similar instrument search functionality to the RSearch function in Thomson Reuters Eikon Excel. The API is accessible via VBA programming in Thomson Reuters Eikon Excel.

This online help uses a coding example to show you how you can work with RSearch COM API.

See also

RSearch function

Using RSearch API in local mode

There are some limitations to using RSearch APIs when Thomson Reuters Eikon Excel is operating in local mode. See "Working in Local Mode" on page 154.

Setting Up RSearch COM API in VBA

Overview

To use RSearch COM API in the VBA environment in Thomson Reuters Eikon Excel, you must first set it up.

How to set up RSearch COM API in VBA

1 Open Thomson Reuters Eikon Excel.

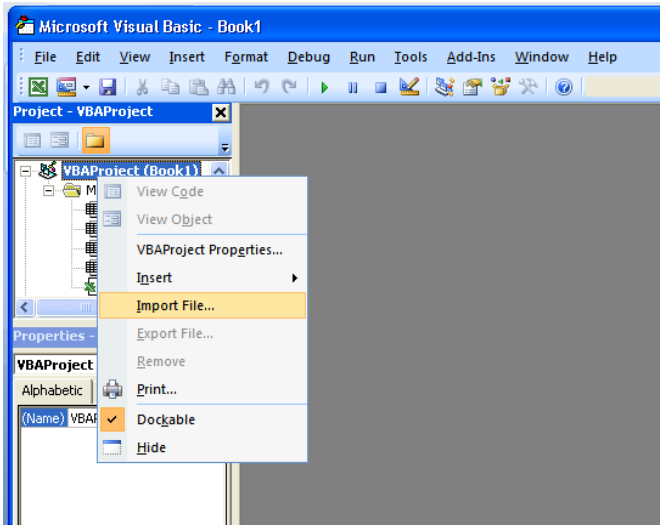
2 Click *Developer* > *Visual Basic* OR press *Alt+F11*.

The VBA screen opens.

The Developer tab is hidden in Microsoft Excel 2007. To display this tab, see "How to display the Developer tab" on page 149"How to display the Developer tab" on page 149.

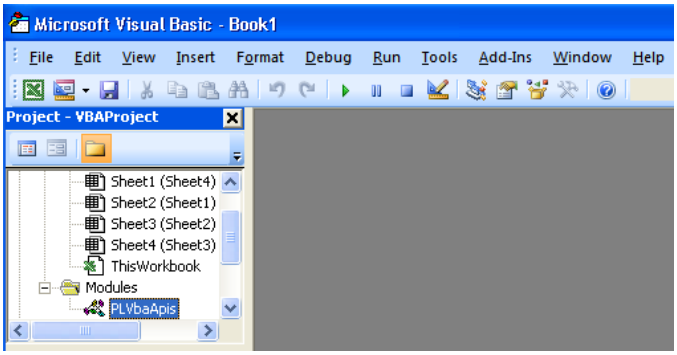
3 Right-click *VBAProject (Bookn)* and choose *Import File*.

The *Import File* window opens.



4 Select *PLVbaApis.bas* from the Thomson Reuters Eikon installation folder and click Open.

The module is added to the VBA project.

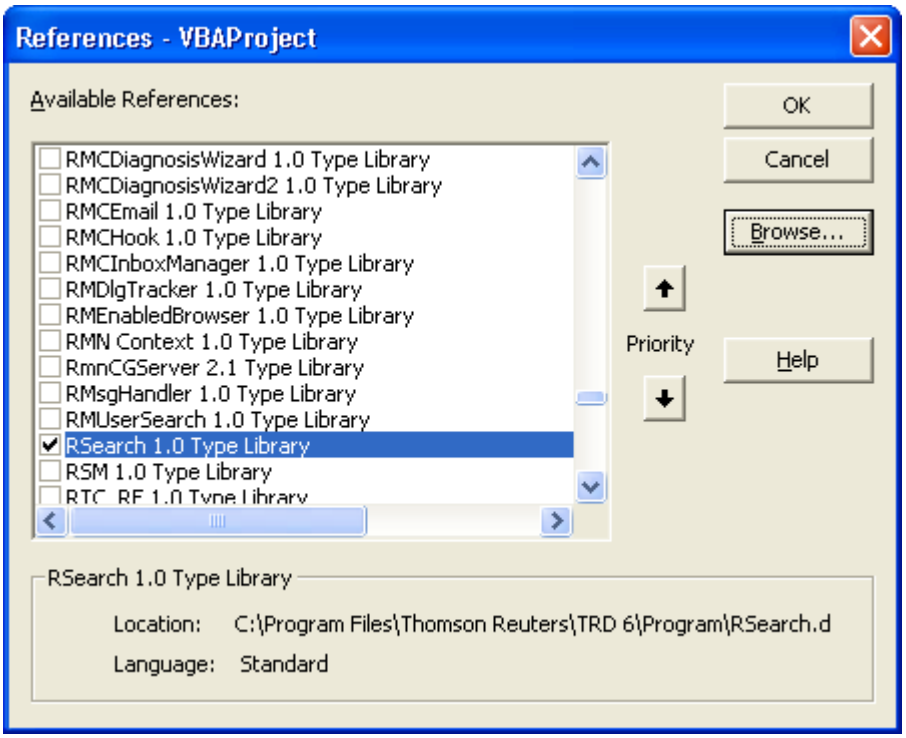


The default paths are `C:\Program Files (x86)\Thomson Reuters\Eikon\X\Bin` and `C:\Program Files (x86)\Thomson Reuters\Eikon\Z\Bin\Apps\TR.OFFICE.CORE\0.0.0.0\Bin` for DEX2 API.

5 Choose *Tools* > *References* > *Browse*.

6 Select the *RSearch.dll* file from the Thomson Reuters Eikon installation folder and click *Open*.

A reference is added to the RSearch library.



7 Repeat steps 5 and 6 to add these files:

- dex2.dll
- rtx.dll
- adxfo.dll
- adxoo.dll

How to display the Developer tab

In Microsoft Excel 2007, the *Developer* tab is hidden by default. You can follow these steps to display the *Developer* tab.

1 Open Microsoft Excel.

2 Click 

3 Click the *Excel Options* button.

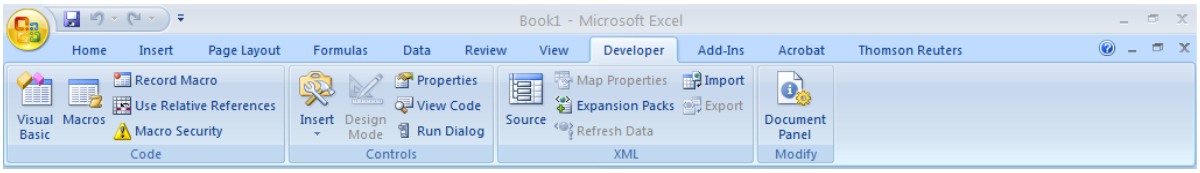
Excel Options opens.

4 Choose *Popular*.

5 Select *Show Developer tab in the Ribbon*.

6 Click *OK*.

The *Developer* tab is displayed.



Creating and Releasing RSearch Manager

VBA sample

This code shows how you can instantiate, initialize, finalize, and release the RSearch Manager object.

```
' Global Variable that holds the instance of the RSearch manager singleton
Dim MyRSearchMgr As RSearchLib.RSearchMgr

' Global variable that holds the cookie that identifies the RSearch session
Dim MyRSearchCookie As Long

Private Sub Worksheet_Activate()
    ' try to instantiate the RSearch manager
    Set MyRSearchMgr = CreateRSearchMgr()

    If Not MyRSearchMgr Is Nothing Then
    'Initialize RSearch session (we do not provide any logger here)
    MyRSearchCookie = MyRSearchMgr.Initialize(RS_CT_EIKON)
    End If
End Sub

Private Sub Worksheet_Deactivate ()
    If Not MyRSearchMgr Is Nothing Then
           ' Release the RSearch session
            MyRSearchMgr.Finalize (MyRSearchCookie)

         ' Release the RSearch manager
         Set MyRSearchMgr = Nothing
    End If
End Sub
```

Creating a Query

VBA sample

This code shows how to prepare and send a query.

```
' Global Variable that holds the instance of the RSearch manager singleton
Dim MyRSearchMgr As RSearchLib.RSearchMgr
' Global variable that holds the cookie that identifies the RSearch session
Dim MyRSearchCookie As Long
' Global variable that holds the RSearch query
Dim WithEvents MyRSearchQuery As RSearchLib.RSearchQuery


Private Sub Worksheet_Activate()
    ' try to instantiate the RSearch manager
    Set MyRSearchMgr = CreateRSearchMgr()

    If Not MyRSearchMgr Is Nothing Then
      ' Initialize RSearch session (we do not provide any logger here)
      MyRSearchCookie = MyRSearchMgr.Initialize(RS_CT_EIKON)

      ' Create a RSearch query using the session cookie
      Set MyRSearchQuery = MyRSearchMgr.CreateRSearchQuery(MyRSearchCookie)

      If Not MyRSearchMgr Is Nothing Then
        'Initialize the RSearch query with the criteria to use for the
search
        MyRSearchQuery.AssetClass = "Equity"
        MyRSearchQuery.SearchCriteria = "EPS:>5
RCSIssuerCountryLeaf:Canada"
        MyRSearchQuery.SearchParameters = "NBROWS:5 SORT:EPS:A"


        ' Send the query
        MyRSearchQuery.Send
      End If
    End If
End Sub
```

Handling a Query Event

VBA sample

Use the OnUpdate event to handle the result of the query. This code shows how to retrieve and display the query results.

```
Private Sub MyRSearchQuery_OnUpdate (ByVal pIRSearchResponse As
RSearchLib.IRSearchResponse)
   ' Check the status of the query
    If pIRSearchResponse.Status = 0 Then
        ' Retrieve the list of instruments
        Dim MyRSearchInstrumentsList As RSearchLib.IRSearchInstrumentsList
        Set MyRSearchInstrumentsList = pIRSearchResponse.Instruments

          ' Display the result starting at A1
           Dim Instruments As RSearchLib.IRSearchInstrument
           For RowIndex = 1 To MyRSearchInstrumentsList.Count
           Set Instrument = MyRSearchInstrumentsList (RowIndex)
           Dim RowIndexStr As String
           RowIndexStr = RowIndex
           Me.Range ("A" + RowIndexStr) = Instrument.Code
        Next
    End If
End Sub
```

Testing the Sample Code

How to test the sample code

1 In the Microsoft Excel toolbar, click *Thomson Reuters* > *Sign In*.
The Thomson Reuters Eikon authentication screen opens.
2 Enter your user ID and password.
Your user ID is usually your e-mail address.
3 Click *Sign In*.
You are signed in to Thomson Reuters Eikon Excel.
4 Click on *Sheet1* and *Sheet2* in succession.
The results of the RSearch function are updated in the cells.

Working in Local Mode

Overview

Local mode is the offline mode of Thomson Reuters Eikon Excel where the application runs without connecting to the Thomson Reuters platform. In this mode, you can only view real-time data from locally deployed feeds.

Activation of local mode

If you are unable to connect to the Thomson Reuters platform, you can sign in to Thomson Reuters Eikon Excel in local mode. If you are already connected to the Thomson Reuters platform and you get disconnected, Thomson Reuters Eikon Excel automatically switches to local mode.

RSearch API usage in local mode

In local mode, RSearch objects can be instantiated normally. However, if any of their methods is called, Thomson Reuters Eikon Excel displays an error. The methods can be used once you reconnect to the Thomson Reuters platform.

RHistory API

Overview

The RHistory API enables developers to access time series data in VBA inside Thomson Reuters Eikon Excel in the same way as the RHistory function.

The RHistory function retrieves a list of time series data for one instrument or a list of instruments at regular intervals (for example, on a daily, weekly, monthly, and yearly basis) for a given time period or for a given number of records. It also provides time series data at non-regular intervals, for example, TAS (Time and Sales), TAQ (Trade and Quote), and TICK (tick by tick).

For more information on the RHistory function, refer to the online help on Eikon Excel.

The RHistory API only supports VBA language and only works in Thomson Reuters Eikon Excel.

"Quick Start" on page 155

"Basic VBA Sample" on page 158

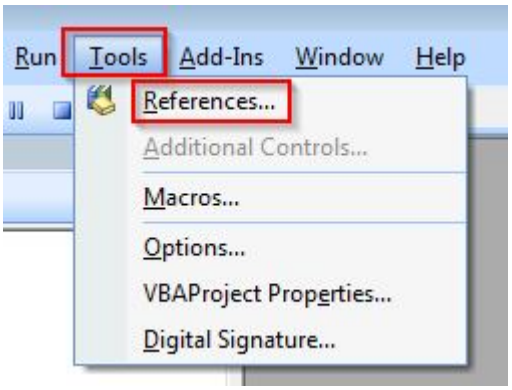
Samples

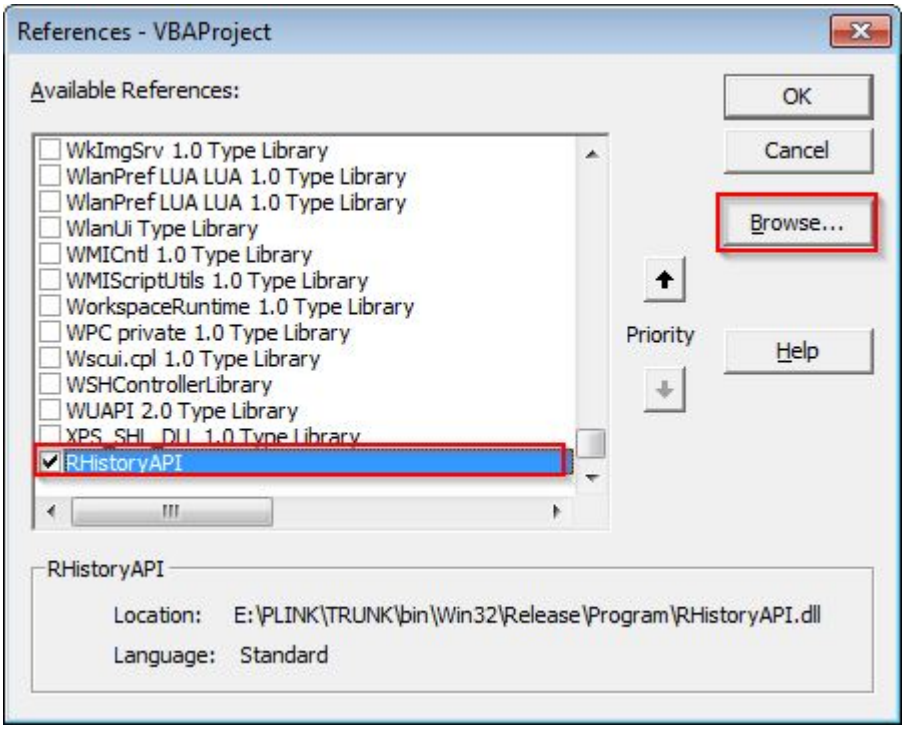
For tutorials including samples, visit the Thomson Reuters Developer Community Portal.

Quick Start

The following sequence illustrates how to use the public RHistory API in VBA in Excel.

1 Add a reference to the type library RHistoryAPI.





2 Declare the required variables.

```
Private m_rhistoryManager As RHistoryAPI.RHistoryManager
Private m_rhistoryCookie As Long
Private WithEvents m_rhistoryQuery As RHistoryAPI.RHistoryQuery
```

3 Declare and wrap the factory function CreateReutersObject provided by Thomson Reuters.

```
Private Declare Function CreateReutersObject Lib "PLVbaApis.dll" (ByVal progID
As String) As Object

Private Function CreateHistoryManager() As RHistoryAPI.RHistoryManager
    Set CreateHistoryManager = CreateReutersObject
("RHistoryAPI.RHistoryManager")
End Function
```

4 Create and initialize a manager and a query when the book is open.

```
Private Sub Workbook_Open()
    Set m_rhistoryManager = CreateHistoryManager
    m_rhistoryCookie = m_rhistoryManager.Initialize("MY BOOK")
    Set m_rhistoryQuery = m_rhistoryManager.CreateHistoryQuery(m_
rhistoryCookie)
End Sub
```

5 Destroy the query and the manager when the book is closed.

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
    Set m_rhistoryQuery = Nothing
    m_rhistoryManager.Terminate (m_rhistoryCookie)
    m_rhistoryCookie = 0
    Set m_rhistoryManager = Nothing
End Sub
```

6 Launch the query on demand in a macro.

```
Public Sub LaunchQuery()
    ' TODO: Set the query parameters accordingly to your needs
    m_rhistoryQuery.InstrumentIdList = "MSFT.O"
    m_rhistoryQuery.FieldList = "TRDPRC_1.TIMESTAMP;TRDPRC_1.HIGH;TRDPRC_
1.CLOSE;TRDPRC_1.LOW;TRDPRC_1.OPEN;TRDPRC_1.VOLUME;TRDPRC_1.COUNT"
    m_rhistoryQuery.RequestParams = "INTERVAL:1M"
    m_rhistoryQuery.RefreshParams = ""
    m_rhistoryQuery.DisplayParams = "CH:In;Fd"
    On Error GoTo ErrorHandler
    m_rhistoryQuery.Subscribe
ErrorHandler:
    ' TODO: Handle the possible synchronous error returned by the method
Subscribe
End Sub
```

7 Cancel the query on demand in a macro.

```
Public Sub CancelQuery()
    m_rhistoryQuery.Cancel
End Sub
```

8 Handle the possible events received by the query.

```
Private Sub m_rhistoryQuery_OnImage(ByVal a_historyTable As Variant)
    ' TODO: Use the data in the array a_historyTable
End Sub

Private Sub m_rhistoryQuery_OnUpdate(ByVal a_historyTable As Variant, ByVal a_
startingRowIndex As Long, ByVal a_startingColumnIndex As Long, ByVal a_
shiftDownExistingRows As Boolean)
    ' TODO: Use the data in the array a_historyTable
End Sub

Private Sub m_rhistoryQuery_OnError(ByVal a_status As Long, ByVal a_
statusDescription As String)
    ' TODO: Handle the possible asynchronous error
End Sub
```

Basic VBA Sample

This basic sample is a canvas to show you a simple usage of the API.

```
Private m_rhistoryManager As RHistoryAPI.RHistoryManager
Private m_rhistoryCookie As Long
Private WithEvents m_rhistoryQuery As RHistoryAPI.RHistoryQuery


Private Declare Function CreateReutersObject Lib "PLVbaApis.dll" (ByVal progID
As String) As Object

Private Function CreateHistoryManager() As RHistoryAPI.RHistoryManager
    Set CreateHistoryManager = CreateReutersObject
("RHistoryAPI.RHistoryManager")
End Function


Private Sub Workbook_Open()
    Set m_rhistoryManager = CreateHistoryManager
    m_rhistoryCookie = m_rhistoryManager.Initialize("MY BOOK")
    Set m_rhistoryQuery = m_rhistoryManager.CreateHistoryQuery(m_
rhistoryCookie)
End Sub


Private Sub Workbook_BeforeClose(Cancel As Boolean)
    Set m_rhistoryQuery = Nothing
    m_rhistoryManager.Terminate (m_rhistoryCookie)
    m_rhistoryCookie = 0
    Set m_rhistoryManager = Nothing
End Sub


Public Sub LaunchQuery()
    ' TODO: Set the query parameters accordingly to your needs
    m_rhistoryQuery.InstrumentIdList = "MSFT.O"
    m_rhistoryQuery.FieldList = "TRDPRC_1.TIMESTAMP;TRDPRC_1.HIGH;TRDPRC_
1.CLOSE;TRDPRC_1.LOW;TRDPRC_1.OPEN;TRDPRC_1.VOLUME;TRDPRC_1.COUNT"
    m_rhistoryQuery.RequestParams = "INTERVAL:1M"
    m_rhistoryQuery.RefreshParams = ""
    m_rhistoryQuery.DisplayParams = "CH:In;Fd"
    On Error GoTo ErrorHandler
    m_rhistoryQuery.Subscribe
ErrorHandler:
    ' TODO: Handle the possible synchronous error returned by the method
Subscribe
End Sub
```

```
Public Sub CancelQuery()
    m_rhistoryQuery.Cancel
End Sub


Private Sub m_rhistoryQuery_OnImage(ByVal a_historyTable As Variant)
    ' TODO: Use the data in the array a_historyTable
End Sub


Private Sub m_rhistoryQuery_OnUpdate(ByVal a_historyTable As Variant, ByVal a_
startingRowIndex As Long, ByVal a_startingColumnIndex As Long, ByVal a_
shiftDownExistingRows As Boolean)
    ' TODO: Use the data in the array a_historyTable
End Sub


Private Sub m_rhistoryQuery_OnError(ByVal a_status As Long, ByVal a_
statusDescription As String)
    ' TODO: Handle the possible asynchronous error
End Sub
```

RHistoryManager Class Reference

General description

Entry-point object that acts as a factory of queries. You should usually create and re-use one object of this type during the lifetime of your application. Then you should use it to create any number of historical query objects. The classic workflow is as follows:

- On application start-up: create a RHistoryManager object, initialize it, and store the returned cookie.
- During the lifetime of your application: create historical queries with the RHistoryManager object.
- Upon exiting your application: call the Terminate method of the RHistoryManager object, passing the stored cookie as a parameter.

RHistoryManager methods

RHistoryManager Initialize

Initialize the history manager and returns a unique cookie that should be given to Terminate when finishing your application. You can call Initialize several times, it will return each time a different unique cookie. You have to call Terminate as much time as you called Initialize and with all the cookies you got.

Arguments

`a_applicationName`	Unique string to identify your application

Return value

A new cookie, to be used when calling Terminate or CreateHistoryQuery.

RHistoryManager CreateHistoryQuery

Factory method to create a RHistoryQuery object.

Arguments

`a_cookie`	Cookie obtained by a call to Initialize

Return value

An RHistoryQuery object.

RHistoryManager Terminate

Terminate the history manager and release any allocated related resources.

Arguments

`a_cookie`	Cookie obtained by a call to Initialize

Return value

None

RHistoryQuery Class Reference

General description

Object representing a historical data query. Only a `RHistoryManager` object can create such objects.

- First, set the query's parameters: list of instrument IDs, list of fields, additional parameters, and display parameters.
- Then, call Subscribe.

RHistoryQuery properties

Tip: Use the RHistory function in TR Eikon – MS Office to get details on the properties for any given instrument.

Property	Description
`InstrumentIdList`	Set a list of instrument IDs, separated by commas, for which you want to query historical data
`FieldList`	Set the list of fields for which you want to query historical data.
`RequestParams`	Set additional parameters for the query
`RefreshParams`	Set the refresh parameters for the query
`DisplayParams`	Set the display-related parameters for the query

RHistoryQuery methods

RHistoryQuery Subscribe

Starts or restarts a query. Query results are sent through RHistoryQuery events.

Arguments

None

Return value

None

RHistoryQuery Cancel

Cancels a query.

Arguments

None

Return value

None

RHistoryQuery Events

General description

Event interface to receive the results of `RHistoryQuery` (a historical query).

Public member functions

HRESULT OnImage ([in] VARIANT *a_historyTable*)

Event triggered when the system sends the initial current historical data.

Arguments

a_historyTable: A two-dimensional array of historical data

HRESULT OnUpdate ([in] VARIANT *a_historyTable*, [in] LONG *a_startingRowIndex*, [in] LONG *a_startingColumnIndex*, [in] VARIANT_BOOL *a_shiftDownExistingRows*)

Event triggered when the system sends real-time updates.

Arguments

a_historyTable: A two-dimensional array of historical data

a_startingRowIndex: Index of the row at which the update should apply

a_startingColumnIndex: Index of the column at which the update should apply

a_shiftDownExistingRows: Indicates whether the existing rows in the image should be shifted down or replaced by the update

HRESULT OnError ([in] LONG *a_status*, [in] BSTR a_statusDescription)

Event triggered when the system sends an error.

Arguments

a_status: Integer code related to the error

a_statusDescription: Textual description of the error

AdfinX Analytics 3.0 Object Library

Introduction

Overview

AdfinX Analytics 3.0 Object Library is a flexible and powerful tool for working with any type of financial instrument from all countries. It has been designed to cover the needs of all bond, forex and equity market professionals.

The AdfinX Analytics 3.0 Object library (AdfinXAnalyticsObjects) provides a standard way of performing financial calculations within Microsoft Windows® applications using Microsoft ActiveX® technology. Tasks that may be accomplished with this module include managing:

- Fixed income instruments (bonds, FRNs, convertible bonds...)
- Interest forex based on all types of references
- Swaps instruments
- Options instruments (equity or interest rate derivatives)
- Exotic options instruments (barrier, rainbow, lookback, Asian...)
- Interest rate models (yield to maturity, zero-coupon curve, Vasicek-Fong, Black, Derman, and Toy, Hull and White, etc.)
- Risk models for credit default swaps

To support these functionalities within a program, the AdfinXAnalyticsObjects library provides objects based on ActiveX interfaces. These are implemented by the *adxoo.dll* file.

Usage

Registration

The first step in being able to use the AdfinX Analytics 3.0 Object Library is to make sure that it has been installed correctly. Essentially, this means that the component identifiers are declared in the Windows NT registry. If you need to re-register the object library, delivered as an in-process DLL called *Adxoo.dll*, use the following command:

```
path-to-file\-regserver
```

Visual Basic®

To use the components from the Visual Basic IDE, use the *Project Components* menu item, and, in the *Controls* property page tab of the dialog box, select the check box for *AdfinX Analytics 3.0 Object Oriented Library*. This will add the AdfinX Analytics objects to your component toolbox. You can also find them with the Object Browser.

VBA (Microsoft Excel®)

On the *Tools* menu, click *References* and then select *AdfinX Analytics 3.0 Object-Oriented Library*. Click *OK*. This will add the AdfinX Analytics objects to your component toolbox. You can also find them with the Object Browser.

C++ (Microsoft Visual C++®)

Use the `#import` directive to import the AdfinX Analytics 3.0 Object Library:

```
#import "adxoo.dll"
```

Hierarchy of objects and interfaces

IAdxObject interface

The majority of AdfinX Analytics interfaces are derived from the IAdxObject interface. This interface provides a status of the object through the error properties and the ability to set and get attributes on the object through its methods. This IAdxObject interface inherits from the IDispatch interface that gives access and control to the properties and methods of the object. ActiveX clients, such as Visual Basic®, can access attributes and errors of AdfinX Analytics objects by invoking properties and methods of the IAdxObject interface.

Instantiation

The AdfinX Analytics Objects act like other COM components when it comes to instantiation: to create an instance of one of the AdfinX Analytics objects you must use the CLSID (class identifier). This is done behind the scenes for you when using Visual Basic: either simply instantiate an object on a form or declare an object of the correct type at the appropriate level, like this:

```
Dim MyAdxAsset As AdfinXAnalyticsObjects.AdxAsset
...
Set MyAdxAsset = new AdfinXAnalyticsObjects.AdxAsset
```

Manipulating the objects

Three main types of AdfinX objects are provided:

Models

Different categories of models can be used:

- rate model: specifies different types of yield curves (Constant, Zero-Coupon, and a yield curve with its volatility)
- risk model: to build a risky zero-coupon curve using the Cox, Ingersoll, and Ross model, a default probability curve, or a constant default probability
- volatility model: the AdfinX Analytics 3.0 Object Library provides a constant volatility curve
- dividend model: specifies the expected value of asset dividends

The first step in using Model objects is to set up their attributes with the IAdxObject Interface: SetAttribute method. The main attributes of the AdxRateModel object are:

- the model type (Zero Coupon, Yield To Maturity, Black, Derman and Toy, Hull and White)
- the day count basis

- the rate type (Actual, Money Market, Continuous Rates)
- the compounding frequency

Instruments

The AdfinX Analytics 3.0 Object Library provides different financial instruments:

- fixed income instruments (bonds, convertibles, swaps, FRNs, etc.)
- credit default swaps
- options (Asian, rainbow, binary, lookback, chooser, etc.)
- other products, such as FRAs, futures, and currencies.

The first step in using objects supporting the IAdxInstrument interface is to set up their properties with the IAdxObject Interface: SetAttribute method. Among all the properties, these objects share a set of properties common to all the financial products (Maturity, Settlement Date, Calendar, and Currency).

The second step is to define the model used to value/price the product, with the IAdxInstrument Interface: AttachModel method. The models have been defined as described in the preceding section.

The third step is to use the IAdxInstrument Interface: AskToCompute method to specify the attributes to be computed.

Calculation Methods

The AdfinX Analytics 3.0 Object Library provides different calculation methods:

- formula-based methods
- tree-based methods
- finite differences

The first step when using the AdxCalcMethod objects is to set up the attributes with the IAdxObject Interface: SetAttribute method.

Once the method has been chosen, the instrument on which you want to do the computation must be attached to the calculation method, with the IAdxInstrument Interface: AttachInstrument method.

The third step is to call the IAdxCalcMethod Interface: Compute method to perform calculations.

The fourth step is to retrieve the calculated values with the IAdxObject Interface: GetAttribute method on the AdxInstrument.

AdfinX Analytics Interfaces

IAdxObject Interface

The `IAdxObject` interface is the base for all interfaces. This interface provides a status of the object through the error properties and the ability to set and get attributes of the object through its methods. This `IAdxObject` interface inherits from the `IDispatch` interface that gives access and control to the properties and methods of the object.

Properties and Methods

See also

- "IAdxInstrument Interface" on page 170
- "IAdxCalcMethod Interface" on page 172
- "IAdxModelBuilder Interface" on page 173
- "IAdxModelBuilder Interface: CreateVolatilityModel" on page 176
- "IAdxDefault Interface" on page 193
- "IAdxTermStructure Interface" on page 221
- "IAdxRiskModel Interface" on page 213
- "IAdxModel Interface" on page 173
- "IAdxParse Interface" on page 210

IAdxObject Interface: ErrorCode

```
ErrorCode() As Long
```

This read-only property is set to an error number if the object encounters an error condition. Under normal conditions, where no error has been encountered, the value of the ErrorCode property is zero.

Arguments

None

See also

"AdxErrorMode" on page 364

IAdxObject Interface: ErrorMode

```
ErrorMode() As AdxErrorMode
```

You can set this property to define the behavior of the object when an error condition arises. According to the value of this property, either an exception is raised (`EXCEPTION`), a dialog box is displayed indicating that an error has occurred (`DIALOGBOX`), or nothing is done (`NO_EXCEPTION`), in which case your code must check for errors.

Arguments

None

See also

"AdxErrorMode" on page 364

IAdxObject Interface: ErrorString

```
ErrorString() As String
```

This property retrieves the error string.

Arguments

None

IAdxObject Interface: AppendStructure

```
AppendStructure(iStructure As String)
```

Arguments

`iStructure`

IAdxObject Interface: GetAttribute

```
GetAttribute(iAttributeType As Variant) As Variant
```

Use this method to retrieve the value of the attribute specified by the `iAttributeType` identifier, available in the enumerated types (see "AdfinX Analytics Parameters and Constants" on page 295).

Arguments

`iAttributeType`	The identifier of the retrieved attribute value

IAdxObject Interface: ResetErrorCode

```
ResetErrorCode()
```

Use this method to reset the error code.

Arguments

None

IAdxObject Interface: SetAttribute

```
SetAttribute(iAttributeType As Variant, Val As Variant) As Long
```

Use this method to set the attribute specified by the `iAttributeType` identifier.

Arguments

`iAttributeType`	A `Variant` value that identifies the type of the attribute retrieved
`Val`	A `Variant` value that specifies the attribute value

IAdxInit

The `IAdxInit` interface is used to overload the Adfin default parameter.

Properties

No properties are available.

Methods

The following methods are:

- "IAdxInit Interface: Enable Multithread" on page 170
- "IAdxInit Interface: Reload" on page 170

See also

- "AdfinX Analytics Objects" on page 228

IAdxInit Interface: Enable Multithread

```
EnableMultiThread(iEnable As Boolean)
```

Use this method to set if you want use Adin in a multithreaded context or not.

Arguments

```
iEnable
```

IAdxInit Interface: Reload

```
Reload()
```

Use this method to reload the default Reload parameter of Adfin.

Arguments

None

IAdxInstrument Interface

The `IAdxInstrument` interface gathers the properties and methods common to the objects that describe the financial instruments available in AdfinX Analytics 3.0 Component Library (e.g. FRNs, Bonds, Convertibles, etc.).

Properties

According to the interface hierarchy, the `IAdxInstrument` interface inherits the `IAdxObject` properties, (see "IAdxObject Interface" on page 167).

Methods

The IAdxInstrument interface also inherits the IAdxObject methods (see "IAdxObject Interface" on page 167). Further methods proper to IAdxInstrument are:

- "IAdxInstrument Interface: AskToCompute" on page 170
- "IAdxInstrument Interface: AttachModel" on page 171
- "IAdxInstrument Interface: AttachInstrument" on page 171
- "IAdxInstrument Interface: GetInstrument" on page 171
- "IAdxInstrument Interface: GetModel" on page 171

See also

- "IAdxObject Interface" on page 167
- "AdxAttrInstrument" on page 321

IAdxInstrument Interface: AskToCompute

```
AskToCompute(iValueId As Long) As Long
```

This method is used to set up the instrument attributes to calculate. These attributes are called “Level #3 Attributes” and prefixed ADX_ATTR3, (see "AdfinX Analytics Parameters and Constants" on page 295).

Arguments

`iValueId`	The attribute identifier

IAdxInstrument Interface: AttachModel

```
AttachModel(iPtrId As Long,ipModel As IAdxModel)
```

Use this method to attach a rate/volatility/dividend model to a financial product, before beginning an attribute retrieval or computation. It is particularly useful for setting the discount rate model used in the pricing of fixed income instruments.

Arguments

`iPtrId`	The model identifier
`ipModel`	A pointer addressed to an AdxModel object

IAdxInstrument Interface: AttachInstrument

```
AttachInstrument(iPtrId As Long,ipInstrument As IAdxInstrument)
```

Use this method to attach a financial product to an instrument. It is particularly useful for setting the underlying instrument of a future contract or an option.

Arguments

`ipInstrument`	A pointer adressed to an object supporting the `IAdxInstrument` interface
`iPtrId`	The instrument identifier

See also

- "IAdxInstrument Interface: GetInstrument" on page 171
- "IAdxInstrument Interface: GetModel" on page 171
- "IAdxInstrument Interface: AskToCompute" on page 170
- "IAdxInstrument Interface: AttachModel" on page 171

IAdxInstrument Interface: GetInstrument

```
GetInstrument(iPtrId As Long) As IAdxInstrument
```

Use this method to retrieve the reference to an instrument that has been attached.

Arguments

`iPtrId`	The instrument identifier

IAdxInstrument Interface: GetModel

```
GetModel(iPtrId As Long) As IAdxModel
```

This method is used to retrieve the reference to a rate/volatility/dividend model.

Arguments

`iPtrId`	The model identifier

IAdxCalcMethod Interface

The `IAdxCalcMethod` interface represents objects that perform calculations according to various methods (formula, tree, and finite difference methods).

Properties

According to the interface hierarchy, the `IAdxCalcMethod` interface inherits the `IAdxObject` properties, (see "IAdxObject Interface" on page 167).

Methods

The `IAdxCalcMethod` interface also inherits the `IAdxObject` methods, (see "IAdxObject Interface" on page 167). Further methods proper to `IAdxCalcMethod` are:

- "IAdxCalcMethod Interface: AttachInstrument" on page 172
- "IAdxCalcMethod Interface: Compute" on page 172
- "IAdxCalcMethod Interface: Init" on page 172
- "IAdxCalcMethod Interface: GetInstrument" on page 173

See also

- "AdxAttrCalcMethod" on page 301
- "AdxCalcMethod" on page 238

IAdxCalcMethod Interface: AttachInstrument

`AttachInstrument(ipInstrument As IAdxInstrument)`

Use this method to attach an instrument to a calculation method in order to perform calculations on its attributes.

Arguments

`ipInstrument`	Identifies the instrument to attach as an `AdxInstrument` object

IAdxCalcMethod Interface: Compute

`Compute()`

Use this method to perform a calculation (valuation or pricing) on an instrument, attached beforehand to the chosen calculation method. Calculated attributes are set using the `AskToCompute` method.

Arguments

None

IAdxCalcMethod Interface: Init

`Init(iStructure As BSTR)` As Long

Use this method to create an `IAdxCalcMethod` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object

IAdxCalcMethod Interface: GetInstrument

```
GetInstrument () As IAdxInstrument
```

Once a computation on attributes of an instrument has been performed, this method is used to detach the instrument from the calculation method.

Arguments

None

IAdxModel Interface

The `IAdxModel` interface is the base interface for all pricing models available in AdfinX Analytics 3.0 Component Library.

Properties

According to the interface hierarchy, the `IAdxModel` interface inherits the `IAdxObject` properties, see (see "IAdxObject Interface" on page 167).

Methods

The `IAdxModel` interface also inherits the `IAdxObject` methods, (see "IAdxObject Interface" on page 167). No further methods are provided.

See also

- "IAdxObject Interface" on page 167

IAdxModelBuilder Interface

The `IAdxModelBuilder` interface is the base interface used to build rate and risk models available in AdfinX Analytics 3.0 Component Library.

Properties

According to the interface hierarchy, the `IAdxModelBuilder` interface inherits the `IAdxObject` properties, (see "IAdxObject Interface" on page 167).

Methods

The `IAdxModelBuilder` interface also inherits the `IAdxObject` methods, (see "IAdxObject Interface" on page 167). Further methods proper to `IAdxModelBuilder` are:

- "IAdxModelBuilder Interface: CreateFxModel" on page 174
- "IAdxModelBuilder Interface: CreateModelIVolCube" on page 174
- "IAdxModelBuilder Interface: CreateRateModel" on page 174
- "IAdxModelBuilder Interface: CreateRiskModel" on page 174

IAdxModelBuilder Interface: CreateFxModel

```
CreateFxModel(iFxModel As AdFxModel, iPaidRate As AdxRateModel, iRecRate As AdxRateModel, iCBSArray, iStartDate) As AdxFxModel
```

Use this method to create a FX model.

Arguments

`iFxModel`

`iPaidRate`

`iRecRate`

`iCBSArray`

`iStartDate`

IAdxModelBuilder Interface: CreateModelIVolCube

```
CreateModelIVolCube(fDate As Double, sMarketConvention As String, ContributionParams, RateArray, RateStructure As String) As AdxVolatilityModel
```

Arguments

`fDate`

`sMarketConvention`

`ContributionParams`

`RateArray`

`RateStructure`

IAdxModelBuilder Interface: CreateRateModel

```
CreateRateModel(CreditStructure As BSTR, InstrumentArray As Variant) As IAdxRateModel
```

Use this method to create a rate model to value and price instruments.

Arguments

`RateStructure`	A `String` value that specifies the structure of the rate model you want to create
`InstrumentArray`	A `Variant` value that specifies an array of instruments used to create the rate model

IAdxModelBuilder Interface: CreateRiskModel

```
CreateRiskModel(CreditStructure As BSTR, InstrumentArray As Variant, RateModel As IAdxRateModel) As IAdxRiskModel
```

Use this method to create a risk model using either the Cox, Ingersoll, and Ross model or a default probability curve.

Arguments

`CreditStructure`	A `String` value that specifies the structure of the risk model you want to create
`InstrumentArray`	A `Variant` value that specifies an array of instruments used to create the risk model
`RateModel`	An `AdxRateModel` object that specifies the risk-free rate model used as input to create the risk model

IAdxModelBuilder Interface: CreateRiskyRateModel

```
CreateRiskyRateModel(RateModel As IAdxRateModel, RiskModel As IadxRiskModel,
StartDate As Variant) As IAdxRateModel
```

Use this method to create a risky zero-coupon curve from a risk-free rate model and a risk model.

Arguments

`RateModel`	An `AdxRateModel` object that specifies the risk-free zero-coupon curve
`RiskModel`	An `AdxRiskModel` object that specifies the risk model used (Cox, Ingersoll, and Ross parameters, default probability curve, or constant default probability)
`StartDate`	A `Variant` value used to specify the start date

IAdxModelBuilder Interface: CreateJLTRiskModel

```
CreateJLTRateModel(CreditStructure As BSTR, TransitionMatrix As Variant,
CreditArray As Variant, RateModel As IAdxRateModel) As IAdxRiskModel
```

Use this method to create a risk model using the Jarrow, Lando, and Turnbull method.

Arguments

`CreditStructure`	A `String` value that specifies the structure of the risk model you want to create
`TransitionMatrix`	A `Variant` value that specifies the structure and content of the transition matrix used in the method
`CreditArray`	A `Variant` value that specifies an array of risky zero-coupon curves used to create the risk model

IAdxModelBuilder Interface: CreateVannaVolgaModel

```
CreateVannaVolgaModel(VolatilityStructure As String, VolatilitySurface,
RateModel) As IAdxRateModel
```

Use this method to create a Vanna Volga model.

Arguments

`VolatilityStructure`	List of keywords that describes the convention used for specifying the volatility surface
`VolatilitySurface`	Input of the volatility surface, which can be incomplete or expressed in bp, or the strike can be in moneyness of the ATM Strike.

`RateModel`	RateModel used to calculate the value of the ATM Strike or other required calculations

IAdxModelBuilder Interface: CreateVolatilityModel

```
CreateVolatilityModel(VolatilityStructure As BSTR, VolatilitySurface As Variant, RateModel As IAdxRateModel, Instrument As IAdxInstrument) As IAdxVolatilityModel
```

Use this method to build a volatility surface from various contributed volatility surface conventions.

Arguments

`VolatilitySurface`	Input of the volatility surface, which can be incomplete or expressed in bp, or the strike can be in moneyness of the ATM Strike.
`VolatilityStructure`	List of keywords that describes the convention used for specifying the volatility surface
`RateModel`	RateModel used to calculate the value of the ATM Strike or other required calculations
`Instrument`	Instrument (like Caps) upon which the input volatility applies. It is used to calculate values such as ATM Strike

IAdxAlgorithm Interface

The `IAdxAlgorithm` interface provides methods and properties algorithmic methods to interpolate, run basic financial calculations like rate conversion, Normal function, Historical volatility, and other calculations.

Properties

According to the interface hierarchy, the `IAdxAlgorithm` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167).

Methods

The `IAdxAlgorithm` interface inherits the `IAdxObject` (see "IAdxObject Interface" on page 167.)

See also

IAdxAlgorithm Interface: SurfaceInterpolate

```
SurfaceInterpolate (iValuesToInterpolate As Variant, oInterpolatedValues As Variant, iInterpMethod As AdxAlgorithmInterp, iObc As AdxAlgorithmYesNo, iNd As AdxAlgorithmYesNo) As Boolean
```

Use this method to interpolate points on a surface. Use the `AttachRange` method to apply the algorithm to the surface or curve on which you want to interpolate your value.

Arguments

`iValuesToInterpolate`	A `Variant` value that describes points to interpolate on the surface
`oInterpolatedValues`	A `Variant` value that describes the interpolated values
`iInterpMethod`	An `AdxAlgorithmInterp` enumerate that describes the interpolation method. The default value is natural cubic spline interpolation without extrapolation
`iObc`	An `AdxAlgorithmYesNo` object that specifies whether or not to perform the out of boundary interpolation check. The default value is NO
`iNd`	An `AdxAlgorithmYesNo` object that describes whether or not to perform null date processing. The default value is NO

IAdxAlgorithm Interface: CurveInterpolate

```
CurveInterpolate (ifloatCalcDate As Variant, iColumn As Variant, interpmethod as AdxAlgorithminterp, iObc As AdxAlgorithmYesNo, iNd As AdxAlgorithmYesNo) As Boolean
```

Use this method to interpolate one point on a curve. Use the `GetFloatResult` method to retrieve the result of the interpolation. Use the `AttachRange` method to apply the algorithm to the surface or curve on which you want to interpolate your value.

Arguments

`ifloatCalcDate`	A `Variant` value that specifies the floatCalcDate
`iColumn`	Rate column to interpolate; default value is 2); 2 = Rate; 3 = Volatility; 4 = MeanReversion
`iInterpMethod`	An `AdxAlgorithmInterp` enumerate that describes the interpolation method. The default value is linear interpolation without extrapolation
`iObc`	An `AdxAlgorithmYesNo` object that specifies whether or not to perform the out of boundary interpolation check. The default value is NO
`iNd`	An `AdxAlgorithmYesNo` object that describes whether or not to perform null date processing. The default value is NO

IAdxAlgorithm Interface: RateConv

```
RateConv (iDatDep As Variant, iDatFin As Variant, isTypeConv As BSTR, ifloatTaux As Variant) As Boolean
```

Use this method to convert a rate from one type to another type.

Arguments

`iDatDep`	A `Variant` value that specifies the begin date for the conversion
`iDatFin`	A `Variant` value that specifies the end date for the conversion
`isTypeConv`	A `BSTR` that specifies the conversion you want, using the FROM and TO keywords
`ifloatTaux`	A `Variant` value that is the rate value you want to convert

IAdxAlgorithm Interface: Parse

```
Parse (oResultRange As Variant, isDataString As BSTR, isParseMode As BSTR)
```

This method parses a string a data string formatted in fractions or bid/ask formats. It returns a range that contains the results of the parsing. The default value is a horizontal 2-cell array containing (`Bid: Bid value`, `Ask: Ask value`).

Arguments

`oResultRange`	A `Variant` range that contains the results of the parse operation
`isDataString`	The String to parse
`isParseMode`	An extended argument that defines the parsing mode

IAdxAlgorithm Interface: HistVolatility

```
HistVolatility (OpMode As BSTR) As Boolean)
```

Use this method to estimate the volatility of a financial instrument based on the historical values of its volatility. Use the `AttachRange` method to attach the historical values to the algorithm object.

Arguments

`OpMode`	A `String` that describes calculation and display options

IAdxAlgorithm Interface: AttachRange

```
AttachRange (ipMrvRange As Variant)
```

Use this method to attach the range history volatility and/or the curve or surface used for the interpolation.

Arguments

`ipMrvRange`	A `Variant` range of values that contains the historical volatilities or the curve/surface on which to interpolate

IAdxAlgorithm Interface: GetFloatResult

```
GetFloatResult () As Double
```

Use this method to retrieve the results of the `CurveInterpolate` and `HistVolatility` methods.

Arguments

None

IAdxAlgorithm Interface: Round

```
Round (iValue As Double, iTick As Double, isRoundMode As Double) As Double
```

Use this method to round a number to the nearest tick.

Arguments

`iValue`	The number to be rounded
`iTick`	A value that specifies the rounding tick
`isRoundMode`	An extended argument that defines the type of rounding

IAdxAlgorithm Interface: Normale

```
Normale (iX As Double) As Double
```

Use this method to calculate `y=f(x)` using a normal distribution.

Arguments

`iX`	The value you use with the formula

IAdxAlgorithm Interface: Init

```
Init ()
```

This method must be called to create an `IAdxAlgorithm` object.

Arguments

None

IAdxAsian Interface

The `IAdxAsian` interface provides methods to specify an Asian exotic option. The Asian object properties are listed in the `AdxAttrAsian` enumerated types.

Properties

According to the interface hierarchy, the `IAdxAsian` interface inherits the `IAdxObject` properties, (see "IAdxObject Interface" on page 167) via IAdxInstrument.

Methods

The `IAdxAsian` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the IAdxInstrument methods (see "IAdxInstrument Interface" on page 170). Further methods proper to IAdxAsian are:

- "IAdxAsian Interface: Init" on page 180

See also

- "AdxAsian" on page 229
- "AdxAttrAsian" on page 297

IAdxAsian Interface: Init

```
Init (iStructure As String)
```

This method must be called to create an `IAdxAsian` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object

IAdxAsset Interface

The `IAdxAsset` interface provides methods and properties that describe an asset instrument. The asset object properties are listed in the `AdxAttrAsset` enumerated type.

Properties

According to the interface hierarchy, the `IAdxAsset` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via IAdxInstrument.

Methods

The `IAdxAsset` interface also inherits the `IAdxObject`, (see "IAdxObject Interface" on page 167)) and the `IAdxInstrument` methods (see "IAdxInstrument Interface" on page 170). Further methods proper to IAdxAsset are:

- "IAdxAsset Interface: Init" on page 180

See also

- "AdxAsset" on page 230
- "AdxAttrAsset" on page 297

IAdxAsset Interface: Init

```
Init (iStructure As String)
```

This method must be called to create an `IAdxAsset` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object

IAdxAssetSwap Interface

The `IAdxAssetSwap` interface describes an asset swap.

Properties

According to the interface hierarchy, the `IAdxAssetSwap` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via IAdxInstrument.

Methods

The `IAdxAssetSwap` interface inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the IAdxInstrument methods (see "IAdxInstrument Interface" on page 170). Further methods proper to IAdxAssetSwap are:

"IAdxAssetSwap Interface: Init" on page 181

See also

"AdxAssetSwap" on page 230

IAdxAssetSwap Interface: Init

```
Init(ipLeg As IAdxLeg, ipUnderlying As IAdxLEg, iSettle As Variant, iMaturity As Variant, LegType As AdxLegAttr)
```

You must call this method to create an `IAdxAssetSwap` interface.

Arguments

`ipLeg`	A `Variant` specifying the leg
`ipUnderlying`	A `Variant` specifying the underlying
`iSettle`	A `Variant` specifying the settlement date
`iMaturity`	A `Variant` specifying the maturity date
`LegType`	Optional; the default value is ADX_LEG_RECEIVED; specifies whether paid or received

IAdxBarrierCapFloor Interface

The `IAdxBarrierCapFloor` interface provides methods and properties that represent barrier caps, floors, and collars. The barrier cap and floor object properties are listed in the `AdxAttrBarrierCapFloor` enumerated type.

Properties

According to the interface hierarchy, the `IAdxBarrierCapFloor` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via IAdxInstrument, `IAdxLeg`, `IAdxFloatLeg`, and `IAdxCapFloor`.

Methods

The `IAdxBarrierCapFloor` interface also inherits the `IAdxObject`, (see "IAdxObject Interface" on page 167) and the IAdxInstrument methods (see "IAdxInstrument Interface" on page 170). Further methods proper to IAdxBarrierCapFloor are:

- "IAdxBarrierCapFloor Interface: Init" on page 182
- "IAdxBarrierCapFloor Interface: IsLegFloat" on page 182

See also

- "AdxBarrierCapFloor" on page 230
- "AdxAttrBarrierCapFloor" on page 299

IAdxBarrierCapFloor Interface: Init

```
Init (iStructure As String)
```

This method must be called to create an `IAdxBarrierCapFloor` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object

IAdxBarrierCapFloor Interface: IsLegFloat

```
IsLegFloat () As Boolean
```

Use this method to check whether the current leg is floating.

Arguments

None

IAdxBasket Interface

The `IAdxBasket` interface provides methods and properties that describe a Basket exotic option. The Basket object properties are listed in the `AdxAttrBasket` enumerated type.

Properties

According to the interface hierarchy, the `IAdxBasket` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via IAdxInstrument.

Methods

The `IAdxBasket` interface also inherits the `IAdxObject`, (see "IAdxObject Interface" on page 167) and the IAdxInstrument methods (see "IAdxInstrument Interface" on page 170). Further methods proper to IAdxBasket are:

- "IAdxBasket Interface: AddInstrument" on page 182
- "IAdxBasket Interface: SetCorrelation" on page 183
- "IAdxBasket Interface: SetWeight" on page 183

See also

- "AdxBasket" on page 233
- "AdxAttrBasket" on page 299

IAdxBasket Interface: AddInstrument

```
AddInstrument(iInstrument As IAdxInstrument) As Long
```

Use this method to add an instrument to the basket of underlying assets on which the exotic option basket is based.

Arguments

`iInstrument`	Identifies the added instrument as an `AdxInstrument` object

IAdxBasket Interface: SetCorrelation

```
SetCorrelation(ipCorrMtx As IAdxCorrelation) As Long
```

Use this method to pass the correlation of assets to the basket.

Arguments

`ipCorrMtx`	Identifies the Correlation Matrix as an `IAdxCorrelation` object

IAdxBasket Interface: SetWeight

```
SetWeight(ipWeight As Double) As Long
```

Use this method to pass the vector of weights to the basket.

Arguments

`ipWeight`	A `Double` value that identifies the vector of weights to be passed

IAdxBond Interface

The `IAdxBond` interface provides methods and properties that describe bond instruments. The bond object properties are listed in the `AdxAttrBond` enumerated type.

Properties

According to the interface hierarchy, the `IAdxBond` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via IAdxInstrument, `IAdxLeg` and `IAdxFixedLeg`.

Methods

The `IAdxBond` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the IAdxInstrument methods (see "IAdxInstrument Interface" on page 170). Further methods proper to IAdxBond are:

- "IAdxBond Interface: IsCallablePuttable" on page 183
- "IAdxBond Interface: Init" on page 183

See also

- "AdxBond" on page 235
- "AdxAttrBond" on page 300

IAdxBond Interface: IsCallablePuttable

```
IsCallablePuttable() As Boolean
```

This method is used to determine whether the bond is callable or puttable.

Arguments

None

IAdxBond Interface: Init

```
Init(iStructure As String)
```

This method must be called to create an `IAdxBond` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object

IAdxCalendar Interface

The `IAdxCalendar` interface provides methods and properties to manage financial calendars.

Properties

No properties are available.

Methods

The following methods are:

See also

- "AdxCalendar" on page 225

IAdxCalendar Interface: AddMonths

```
AddMonths(iMonthNumber As Long, iDate As Variant, iEMC As Long, iDMC As Long)
As Variant
```

Use this method to add a given number of months to a date.

Arguments

`iMonthNumber`	A `Long` value that specifies the number of months to add
`iEMC`	A `Long` value that identifies the `AdxEndOfMonthConvention` value
`iDMC`	A `Long` value that identifies the `AdxEndOfMonthConvention` value
`iDate`	Date computed by adding `iMonthNumber` to its input value

IAdxCalendar Interface: AddPeriod

```
AddPeriod(iPeriod As BSTR, iDate As Variant, iEMC As AdxEndOfMonthConvention,
iDMC As AdxDateMovingConvention) As Variant
```

Use this method to add a period (number of calendar days, working days, weeks, months, or years) to a date.

Arguments

`iPeriod`	A `Variant` value that specifies the number of period to add
`iEMC`	Identifies the `AdxEndOfMonthConvention` value
`iDMC`	Identifies the `AdxDateMovingConvention` value
`oDate`	Date computed by adding `iMonthNumber` to its input value

IAdxCalendar Interface: AddWorkingDays

```
AddWorkingDays(iDate As Variant, iNbDays As Long) As Variant
```

Use this method to add a given number of working days to a calendar date. It returns the next holiday date if the number of days is 0. If an error occurs, the return value is “0L”.

Arguments

`iDate`	A `Variant` value that specifies the reference date
`iNbDays`	A `Long` value that specifies the number of days to add to that date

```
AddYears(iYearsNumber As Long, iDate As Variant, iEMC As Long, iDMC As Long)
As Variant
```

Use this method to add a given number of years a date.

Arguments

`iYearsNumber`	The number of years to add to the `iDate`
`iEMC`	Identifies the `AdxEndOfMonthConvention` value
`iDMC`	Identifies the `AdxDateMovingConvention` value
`iopDate`	Date computed by adding `iMonthNumber` to its input value

IAdxCalendar Interface: AdjustDate

```
AdjustDate(iDate As Variant, iBusinessConv As Long) As Variant
```

Use this method to adjust a date of a calendar, using Business Convention. If an error occurs, the return date is set to 0.

Arguments

`iDate`	A `Variant` value that specifies the date to be adjusted
`iBusinessConv`	A `Long` value that specifies the Business Convention Code to apply

IAdxCalendar Interface: GetBasicCalendarCodes

```
GetBasicCalendarCodes (iDate As Variant) As BSTR
```

Use this method to retrieve the basic calendars for which a date is a non-working day. Three cases of return values can occur:

- If the reference date is a holiday, the method returns a string containing Basic Calendar Codes.
- If the reference date is a working day, the method returns an empty string.
- If an error occurs, “NULL” is returned.

Arguments

`iDate`	A `Variant` value that specifies the reference date

IAdxCalendar Interface: GetBasicCalendarCount

```
GetBasicCalendarCount() As Long
```

Use this method to retrieve the number of basic calendars contained in the `AdxCalendar` object. If the object is not valid, 0 is returned.

Arguments

None

IAdxCalendar Interface: GetHolidayCount

```
GetHolidayCount() As Long
```

Use this method to calculate the number of non-working days.

Arguments

None

IAdxCalendar Interface: GetHolidayList

```
GetHolidayList(iStartDate As Variant, iEndDate As Variant, iDisplayText As
Variant_Boolean, iOrientation As Long) As Variant
```

Use this method to list one or more calendar holidays between two dates.

Arguments

`iStartDate`	A `Variant` value that identifies the first reference date
`iEndDate`	A `Variant` value that identifies the second reference date
`iDisplayText`	Array containing holidays between `iStartDate` and `iEndDate` for all listed calendars

IAdxCalendar Interface: Init

```
Init(iStructure As Variant) As Long
```

Use this method to init a calendar through its structure or style.

Arguments

`iStructure`	A `Variant` value that identifies the structure to be passed

IAdxCalendar Interface: IsWorkingDay

```
IsWorkingDay(iDate As Variant) As Variant_Boolean
```

Use this method to check if the passed date is a working day or not.

Arguments

`iDate`	A `Variant` value that specifies the reference date

IAdxCalendar Interface: LastWorkingDayOfMonth

```
LastWorkingDayOfMonth(iDate As Variant) As Variant
```

Use this method to calculate the last working day of a month.

Arguments

`iDate`	A `Variant` value that specifies the reference date

IAdxCalendar Interface: NonWorkingDaysCount

```
NonWorkingDaysCount(iStartDate As Variant, iEndDate As Variant) As Long
```

Use this method to count the number of non-working days between two dates.

Arguments

`iStartDate`	A `Variant` value that identifies the first reference date
`iEndDate`	A `Variant` value that identifies the second reference date

IAdxCalendar Interface: WorkingDaysCount

```
WorkingDaysCount(iStartDate As Variant, iEndDate As Variant) As Long
```

Use this method to count the number of working days between two dates.

Arguments

`iStartDate`	A `Variant` value that identifies the first reference date
`iEndDate`	A `Variant` value that identifies the second reference date

IAdxCapFloor Interface

The `IAdxCapFloor` interface provides methods and properties that represent caps, floors, and collars. The cap/floor object properties are listed in the `AdxAttrCapFloor` enumerated type.

Properties

According to the interface hierarchy, the `IAdxCapFloor` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via IAdxInstrument, `IAdxLeg` and `IAdxFloatLeg`.

Methods

The `IAdxCapFloor` interface also inherits the `IAdxObject`, (see "IAdxObject Interface" on page 167) and the IAdxInstrument methods (see "IAdxInstrument Interface" on page 170). Further methods proper to

IAdxCapFloor are:

- "IAdxCapFloor Interface: Init" on page 188
- "IAdxCapFloor Interface: IsLegFloat" on page 188

See also

- "AdxCapFloor" on page 239
- "AdxAttrCapFloor" on page 302

IAdxCapFloor Interface: Init

```
Init (iStructure As String)
```

This method must be called to create an `IAdxCapFloor` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object

IAdxCapFloor Interface: IsLegFloat

```
IsLegFloat () As Boolean
```

Use this method to check whether the current leg is floating.

Arguments

None

IAdxCashFlow Interface

The `IAdxCashFlow` interface provides methods and properties that describe the cashflow-based instruments. The cashflow object properties are listed in the `AdxAttrCashFlow` enumerated type.

Properties

According to the interface hierarchy, the `IAdxCashFlow` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via IAdxInstrument.

Methods

The `IAdxCashFlow` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the IAdxInstrument methods (see "IAdxInstrument Interface" on page 170). Further methods proper to IAdxCashFlow are:

- "IAdxCashFlow Interface: Init" on page 188

See also

- "AdxCashFlow" on page 241
- "AdxAttrCashFlow" on page 303

IAdxCashFlow Interface: Init

```
Init (iRange As Variant)
```

This method must be called to create an `IAdxCashFlow` object.

Arguments

`iRange`	A `Variant` range of values required to initialize the object

IAdxCDOTranche

The `IAdxCDOTranche` interface provides methods and properties that describe collateralized debt obligations. The CDO object properties are listed in the `AdxAttrCDOTranche` enumerated type.

Properties

According to the interface hierarchy, the `IAdxCDOTranche` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via IAdxInstrument.

Methods

The `IAdxCDOTranche` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the IAdxInstrument methods (see "IAdxInstrument Interface" on page 170). Further methods proper to IAdxCDOTranche are:

- "IAdxChooser Interface" on page 189

See also

- "AdfinX Analytics Objects" on page 228
- "AdfinX Analytics Parameters and Constants" on page 295

IAdxCDOTranche Interface: Init

```
Init(iStructure As String)
```

This method must be called to create an `IAdxCDOTranche` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object

IAdxChooser Interface

The `IAdxChooser` interface provides methods and properties that describe Chooser exotic options. The Chooser object properties are listed in the `AdxAttrChooser` enumerated type.

Properties

According to the interface hierarchy, the `IAdxChooser` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via IAdxInstrument.

Methods

The `IAdxChooser` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the IAdxInstrument methods (see "IAdxInstrument Interface" on page 170). Further methods proper to

IAdxChooser are:

- "IAdxChooser Interface: Init" on page 190
- "IAdxChooser Interface: InitStructure" on page 190

See also

"AdxChooser" on page 244

IAdxChooser Interface: Init

```
InitChooser(iInstrument1 As IAdxInstrument, iInstrument2 As IAdxInstrument) As Long
```

Use this method to initialize the chooser option with two instrument objects passed as parameters.

Arguments

`iInstrument1`	Identifies the first instrument to choose as an `AdxInstrument` object
`iInstrument2`	Identifies the second instrument to choose as an `AdxInstrument` object

IAdxChooser Interface: InitStructure

```
InitStructure(iStructure As String)
```

This method must be called to create an `IAdxChooser` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object

IAdxConvBond Interface

The `IAdxConvBond` interface provides methods and properties to manage convertible bonds. The convertible bond properties are listed in the `AdxAttrConvBond` enumerated type.

Properties

According to the interface hierarchy, the `IAdxConvBond` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via IAdxInstrument, `IAdxLeg`, `IAdxFixedLeg` and `IAdxBond`.

Methods

The `IAdxConvBond` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the IAdxInstrument methods (see "IAdxInstrument Interface" on page 170). Further methods proper to IAdxBond are:

- "IAdxConvBond Interface: Init" on page 191
- "IAdxConvBond: IsCallablePuttable" on page 191

See also

- "AdxConvBond" on page 247
- "AdxAttrConvBond" on page 305

IAdxConvBond Interface: Init

```
Init(iStructure As String)
```

This method must be called to create an `IAdxConvBond` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object

IAdxConvBond: IsCallablePuttable

```
IsCallablePuttable() As Boolean
```

This method is used to determine whether the convertible bond is callable or puttable.

Arguments

None

IAdxCorrelation Interface

The IAdxCorrelation interface provides methods to perform correlation computation.

Properties

According to the interface hierarchy, the IAdxCorrelation interface inherits the IAdxObject properties, (see "IAdxObject Interface" on page 167).

Methods

The IAdxCorrelation interface also inherits the `IAdxObject` methods, (see "IAdxObject Interface" on page 167). Further methods proper to `IAdxCorrelation` are:

- "IAdxCorrelation Interface: Covar" on page 191
- "IAdxCorrelation Interface: Rho" on page 192
- "IAdxCorrelation Interface: Init" on page 192

See also

- "AdxAttrCorrelation" on page 306
- "AdxCorrelation" on page 250

IAdxCorrelation Interface: Covar

```
Covar(i As Long, j As Long) As Variant
```

Use this method to calculate the covariance coefficient at line i, column j.

Arguments

`i`	A `Long` value that specifies the line in the matrix
`j`	A `Long` value that specifies the column in the matrix

IAdxCorrelation Interface: Init

```
Init(iCorrelationArray)
```

This method must be called to create an `AdxCorrelation` object.

Arguments

`iCorrelationArray`	An array of values required to initialize the object

IAdxCorrelation Interface: Rho

```
Rho(i As Long, j As Long) As Variant
```

Use this method to retrieve the correlation coefficient at line i, column j.

Arguments

`i`	A `Long` value that specifies the line in the matrix
`j`	A `Long` value that specifies the column in the matrix

IAdxCrossCurrency Interface

The `IAdxCrossCurrency` interface provides methods and properties to manage cross currency securities.

Properties

According to the interface hierarchy, the `IAdxCrossCurrency` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via IAdxInstrument.

Methods

The `IAdxCrossCurrency` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the IAdxInstrument methods (see "IAdxInstrument Interface" on page 170). Further methods proper to IAdxCrossCurrency are:

- "IAdxCrossCurrency Interface: Init" on page 192

See also

- "AdxCrossCurrency" on page 251
- "AdxAttrCrossCurrency" on page 306

IAdxCrossCurrency Interface: Init

```
Init(iStructure As String)
```

This method must be called to create an `IAdxCrossCurrency` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object

IAdxCurrency Interface

The `IAdxCurrency` interface provides methods and properties to manage currency.

Properties

According to the interface hierarchy, the `IAdxCurrency` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via IAdxInstrument.

Methods

The `IAdxCurrency` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the IAdxInstrument methods (see "IAdxInstrument Interface" on page 170). No further methods are provided.

See also

- "AdxCurrency" on page 252
- "AdxAttrCurrency" on page 308

IAdxDefault Interface

The IAdxDefault interface provides methods to set and get the default values of the AdfinX Analytics components.

Properties

According to the interface hierarchy, the IAdxDefault interface inherits the IAdxObject properties, (see "IAdxObject Interface" on page 167).

Methods

The IAdxDefault interface also inherits the IAdxObject methods, (see "IAdxObject Interface" on page 167). Further methods proper to IAdxDefault are:

- "IAdxDefault Interface: AddStructure" on page 193
- "IAdxDefault Interface: GetName" on page 194
- "IAdxDefault Interface: GetStructure" on page 194
- "IAdxDefault Interface: Load" on page 194
- "IAdxDefault Interface: Save" on page 194
- "IAdxDefault Interface: Init" on page 194

See also

- "IAdxObject Interface" on page 167
- "AdxDefault" on page 253

IAdxDefault Interface: AddStructure

```
AddStructure(iStructure As Variant) As Long
```

Use this method to modify the default values of an object directly by passing it a structure.

Arguments

`iStructure`	A `Variant` value that identifies the structure to be passed

IAdxDefault Interface: GetName

```
GetName() As Variant
```

Use this method to retrieve the name of the default value of an object.

Arguments

None

IAdxDefault Interface: GetStructure

```
GetStructure() As String
```

Use this method to retrieve the structure that describes the default values of an object.

Arguments

None

IAdxDefault Interface: Load

```
Load() As Long
```

Use this method to load the default values required to set the ActiveX objects.

Arguments

None

IAdxDefault Interface: Save

```
Save() As Long
```

Use this method to save the loaded default values.

Arguments

None

IAdxDefault Interface: Init

```
Init(iStructure As String)
```

This method must be called to create an `IAdxDefault` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object

IAdxDigitalCapFloor Interface

The `IAdxDigitalCapFloor` interface provides methods and properties that represent All or Nothing caps, floors, and collars. The digital cap and floor object properties are listed in the `AdxAttrDigitalCapFloor`

enumerated type.

Properties

According to the interface hierarchy, the `IAdxDigitalCapFloor` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via IAdxInstrument, `IAdxLeg`, `IAdxFloatLeg`, and `IAdxCapFloor`.

Methods

The `IAdxDigitalCapFloor` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the IAdxInstrument methods (see "IAdxInstrument Interface" on page 170). Further methods proper to IAdxDigitalCapFloor are:

- "IAdxDigitalCapFloor Interface: Init" on page 195
- "IAdxDigitalCapFloor Interface: IsLegFloat" on page 195

See also

- "AdxDigitalCapFloor" on page 254
- "AdxAttrDigitalCapFloor" on page 309

IAdxDigitalCapFloor Interface: Init

```
Init (iStructure As String)
```

This method must be called to create an `IAdxDigitalCapFloor` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object

IAdxDigitalCapFloor Interface: IsLegFloat

```
IsLegFloat () As Boolean
```

Use this method to check whether the current leg is floating.

Arguments

None

IAdxDividendModel Interface

The `IAdxInstrument` interface provides methods and properties to manage the dividend model used in product valuation/pricing.

Properties

According to the interface hierarchy, the `IAdxDividendModel` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via IAdxModel.

Methods

The `IAdxDividendModel` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the IAdxModel (see "IAdxModel Interface" on page 173) methods. Further methods proper to

`IAdxDividendModel` are:

- "IAdxDividendModel Interface: Init" on page 196

See also

- "AdxDividendModel" on page 257
- "AdxAttrDividendModel" on page 309

IAdxDividendModel Interface: Init

```
Init (iStructure As String)
```

This method must be called to create an `IAdxDividendModel` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object

IAdxFixedLeg Interface

The `IAdxFixedLeg` interface provides methods and properties that represent fixed-leg-based instruments, such as bonds. The fixed-leg object properties are listed in the `AdxAttrFixedLeg` enumerated type.

Properties

According to the interface hierarchy, the `IAdxFixedLeg` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via IAdxInstrument and `IAdxLeg`.

Methods

The `IAdxFixedLeg` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the IAdxInstrument methods (see "IAdxInstrument Interface" on page 170). Further methods proper to IAdxFloatLeg are:

- "IAdxFixedLeg Interface: Init" on page 196

See also

- "AdxFixedLeg" on page 257
- "AdxAttrFixedLeg" on page 310

IAdxFixedLeg Interface: Init

```
Init (iStructure As BSTR)
```

This method must be called to create an `IAdxFixedLeg` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object.

IAdxFloatLeg Interface

The `IAdxFloatLeg` interface provides methods and properties that represent float-leg-based instruments, such as FRNs. The float-leg object properties are listed in the `AdxAttrFloatLeg` enumerated type.

Properties

According to the interface hierarchy, the `IAdxFloatLeg` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via IAdxInstrument and `IAdxLeg`.

Methods

The `IAdxFloatLeg` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the IAdxInstrument methods (see "IAdxInstrument Interface" on page 170). Further methods proper to IAdxFloatLeg are:

"IAdxFloatLeg Interface: Init" on page 197

"IAdxFloatLeg Interface: IsLegFloat" on page 197

See also

- "AdxFloatLeg" on page 257
- "AdxAttrFloatLeg" on page 311

IAdxFloatLeg Interface: Init

```
Init (iStructure As BSTR)
```

This method must be called to create an `IAdxFloatLeg` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object.

IAdxFloatLeg Interface: IsLegFloat

```
IsLegFloat (isFloat As Variant) As Boolean
```

This method returns true if the leg is a floating leg, false if it is a fixed leg.

Arguments

None

IAdxForex Interface

The `IAdxForex` interface provides methods and properties to manage Forex instruments. The Forex object properties are listed in the `AdxAttrForex` enumerated type.

Properties

According to the interface hierarchy, the `IAdxForex` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via IAdxInstrument and `IAdxCrossCurrency`.

Methods

The `IAdxForex` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the `IAdxInstrument` methods (see "IAdxInstrument Interface" on page 170). Further methods proper to IAdxForex are:

- "IAdxForex Interface: Init" on page 198

See also

- "AdxForex" on page 258
- "AdxAttrForex" on page 312

IAdxForex Interface: Init

```
Init (iStructure As String)
```

This method must be called to create an `IAdxForex` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object

IAdxFra Interface

The `IAdxFra` interface provides methods and properties to manage FRA instruments. The FRA object properties are listed in the `AdxAttrFra` enumerated type.

Properties

According to the interface hierarchy, the `IAdxFra` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via `IAdxInstrument`.

Methods

The `IAdxFra` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the `IAdxInstrument` methods (see "IAdxInstrument Interface" on page 170). Further methods proper to `IAdxFra` are:

- "IAdxFra Interface: Init" on page 198
- "AdxFra" on page 258
- "AdxAttrFra" on page 316

IAdxFra Interface: Init

```
Init (iStructure As BSTR)
```

This method must be called to create an `IAdxFra` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object.

IAdxFrn Interface

The `IAdxFrn` interface provides methods and properties to manage floating rate notes. The FRN object properties are listed in the `AdxAttrFrn` enumerated type.

Properties

According to the interface hierarchy, the `IAdxFrn` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via IAdxInstrument, `IAdxLeg` and `IAdxFloatLeg`.

Methods

The `IAdxFrn` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the `IAdxInstrument` methods (see "IAdxInstrument Interface" on page 170). Further methods proper to IAdxFrn are:

- "IAdxFrn Interface: Init" on page 199
- "IAdxFrn Interface: IsCallablePuttable" on page 199

See also

- "AdxFrn" on page 260
- "AdxAttrFrn" on page 317

IAdxFrn Interface: Init

```
Init (iStructure As String)
```

This method must be called to create an `IAdxFrn` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object.

IAdxFrn Interface: IsCallablePuttable

```
IsCallablePuttable() As Variant_Boolean
```

This method is used to determine whether the FRN is callable or puttable.

Arguments

None

IAdxFrq Interface

The `IAdxFrq` interface provides methods and properties for frequency description. The Frq object properties are listed in the `AdxFrequency` and `AdxFrequencyType` enumerated type.

Properties

According to the interface hierarchy, the `IAdxFrq` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167).

Methods

The `IAdxFrq` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167). Further methods proper to `IAdxFrq` are:

- "IAdxFrq Interface: GetFrequency" on page 200
- "IAdxFrq Interface: GetFrequencyType" on page 200
- "IAdxFrq Interface: GetFRQ" on page 200
- "IAdxFrq Interface: Init" on page 200
- "IAdxFrq Interface: SetFrequency" on page 201
- "IAdxFrq Interface: SetFrequencyType" on page 201
- "IAdxFrq Interface: SetFRQ" on page 201

See also

- "AdxFrequency" on page 366
- "AdxFrequencyType" on page 367

IAdxFrq Interface: GetFrequency

```
GetFrequency() As Double
```

Arguments

None

IAdxFrq Interface: GetFrequencyType

```
GetFrequencyType() As AdxFrequencyType
```

This method is used to get the frequency type value.

Arguments

None

IAdxFrq Interface: GetFRQ

```
GetFRQ() As AdxFrequency
```

This method is used to get the compounding frequency value.

Arguments

None

IAdxFrq Interface: Init

```
Init (iStructure As String)
```

This method must be called to create an `IAdxFrq` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object.

IAdxFrq Interface: SetFrequency

```
SetFrequency(iFrequency As Double) As Long
```

Arguments

```
iFrequency
```

IAdxFrq Interface: SetFrequencyType

```
SetFrequencyType(eFreqType As AdxFrequencyType) As Long
```

Arguments

```
eFreqType
```

IAdxFrq Interface: SetFRQ

```
SetFRQ(iFRQAttributeValue As AdxFrequency) As Long
```

Arguments

```
iFRQAttributeValue
```

IAdxFuture Interface

The IAdxFuture interface provides methods and properties to manage futures. The future object properties are listed in the AdxAttrFuture enumerated type.

Properties

According to the interface hierarchy, the IAdxFuture interface inherits the IAdxObject properties (see "IAdxObject Interface" on page 167) via IAdxInstrument.

Methods

The IAdxFuture interface also inherits the IAdxObject (see "IAdxObject Interface" on page 167) and the IAdxInstrument methods (see "IAdxInstrument Interface" on page 170). Further methods proper to IAdxInstrument are:

- "IAdxFuture Interface: Init" on page 201

See also

- "AdxFuture" on page 263
- "AdxAttrFuture" on page 318

IAdxFuture Interface: Init

```
Init (iStructure As String)
```

This method must be called to create an `IAdxFuture` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object

IAdxFxModel Interface

The `IAdxInstrument` interface provides methods and properties to manage the forex model used in product valuation/pricing.

Properties

According to the interface hierarchy, the `IAdxFxModel` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via IAdxModel.

Methods

The `IAdxFxModel` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the `IAdxModel` (see "IAdxModel Interface" on page 173) methods. Further methods proper to `IAdxFloatLeg` are:

- "IAdxFxModel Interface: Init" on page 202
- "IAdxFxModel Interface: CalcRate" on page 202

See also

- "AdxFxModel" on page 266
- "AdxAttrFxModel" on page 319

IAdxFxModel Interface: Init

```
Init(iStructure As String)
```

This method must be called to create an `IAdxFxModel` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object

IAdxFxModel Interface: CalcRate

```
CalcRate(iStartDate As Variant, iEndDate As Variant) As Variant
```

Use this method to retrieve the forward rate between StartDate and EndDate.

Arguments

`iStartDate`	A `Variant` value that specifies the StartDate
`iEndDate`	A `Variant` value that specifies the EndDate

IAdxIdxStyle Interface

The `IAdxIdxStyle` provides methods and properties to manage index styles.

Properties

According to the interface hierarchy, the `IAdxIdxStyle` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via IAdxStyle.

Methods

The `IAdxIdxStyle` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the `IAdxStyle` (see "IAdxStyle Interface" on page 215) methods. Further methods proper to `IAdxIdxStyle` are:

- "IAdxIdxStyle Interface: AddValue" on page 203
- "IAdxIdxStyle Interface: GetValues" on page 203
- "IAdxIdxStyle Interface: GetValuesCount" on page 203
- "IAdxIdxStyle Interface: LoadHistory" on page 204
- "IAdxIdxStyle Interface: AddValue" on page 204
- "IAdxIdxStyle Interface: DeleteValues" on page 204

See also

- "AdxIdxStyle" on page 225

IAdxIdxStyle Interface: AddValue

```
AddValues(iDatesValues As Variant) As Long
```

Use this method to add an element to the style.

Arguments

`iDate`	A `Variant` value that identifies the element date
`iIndex`	A `Variant` value that identifies the element value

IAdxIdxStyle Interface: GetValues

```
GetValues(iStartDate As Variant, iEndDate As Variant, iIndexDates As Variant)
As Variant
```

Use this method to return the index values between two dates.

Arguments

`iStartDate`	A `Variant` value that identifies the first reference date
`iEndDate`	A `Variant` value that identifies the second reference date
`iIndexDates`	The index dates and values

IAdxIdxStyle Interface: GetValuesCount

```
GetValuesCount() As Long
```

Use this method to retrieve the number of elements of the style.

Arguments

None

IAdxIdxStyle Interface: LoadHistory

```
LoadHistory(iHistoryCode As Variant) As Long
```

Use this method to load an index defined by its style code.

Arguments

`iHistoryCode`	Style code of the index

IAdxIdxStyle Interface: DeleteValues

```
DeleteValues(iStartDate As Variant, iEndDate As Variant)
```

Use this method to remove an index value from the database file on a date you specify.

Arguments

`iDate`	A `Variant` value that identifies the element date
`iIndex`	A `Variant` value that identifies the index value

IAdxIdxStyle Interface: AddValue

```
AddValue(iDate As Variant, iValue as Double)
```

Use this method to add an index value to the database file on a date you specify.

Arguments

`iDate`	A `Variant` value that identifies the element date
`iValue`	A `Variant` value that identifies the index value

IAdxIlb Interface

The `IAdxIlb` interface provides methods and properties to manage index-linked bonds. The Ilb object properties are listed in the `AdxAttrIlb` enumerated type.

Properties

According to the interface hierarchy, the `IAdxIlb` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via IAdxInstrument, `IAdxLeg`, `IAdxFixedLeg` and `IAdxBond`.

Methods

The `IAdxIlb` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the IAdxInstrument methods (see "IAdxInstrument Interface" on page 170). Further methods proper to `IAdxIlb` are:

- "IAdxIlb Interface: Init" on page 205
- "IAdxIlb Interface: IsCallablePuttable" on page 205

See also

- "AdxIlb" on page 266
- "AdxAttrIlb" on page 320

IAdxIlb Interface: Init

```
Init (iStructure As String)
```

This method must be called to create an `IAdxIlb` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object

IAdxIlb Interface: IsCallablePuttable

```
IsCallablePuttable() As Boolean
```

This method is used to determine whether the index-linked bond is callable or puttable.

IAdxLeg Interface

The `IAdxInstrument` interface provides methods and properties to manage the leg-based instruments.

Properties

According to the interface hierarchy, the `IAdxLeg` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via IAdxInstrument.

Methods

The `IAdxLeg` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the IAdxInstrument methods (see "IAdxInstrument Interface" on page 170). No further methods are provided.

See also

- "AdxAttrLeg" on page 322
- "AdxDateMovingConvention" on page 357
- "AdxEndOfMonthConvention" on page 363

IAdxLibor Interface

The `IAdxLibor` interface provides methods and properties to define an IndexRateModel used for FRN calculations, such as the 3 month Libor.

Properties

According to the interface hierarchy, the `IAdxLibor` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167).

Methods

The `IAdxLibor` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167). Further methods proper to `IAdxLibor` are:

- "IAdxLibor Interface: AttachModel" on page 206
- "IAdxLibor Interface: GetModel" on page 206
- "IAdxLibor Interface: Init" on page 206

See also

- "AdxLibor" on page 269

IAdxLibor Interface: AttachModel

```
AttachModel(iPtrId As Long, ipModel As IAdxModel)
```

This method is called to attach a rate/volatility/dividend model to a financial product, before starting an attribute retrieval or computation. It is particularly useful to set the discount rate model used in the pricing of fixed income instruments.

Argument

iPtrId	The model identifier
`ipModel`	A pointer adressed to an AdxModel object

IAdxLibor Interface: GetModel

```
GetModel(iName As BSTR)
```

Use this method to retrieve the underlying Libor model of the Libor object.

Argument

`iName`	Name of the Libor object, and also used to handle the Libor object in a MapLibor

IAdxLibor Interface: Init

```
Init (iStructure As String)
```

This method must be called to create an `IAdxLibor` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object

IAdxMapLibor Interface

The `IAdxMapLibor` interface is an stl map of `AdxLibor` object. It is the container that is attached to the FRN to supply all the IndexRateModel information.

Properties

According to the interface hierarchy, the `IAdxMapLibor` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167).

Methods

The `IAdxMapLibor` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the IAdxInstrument methods (see "IAdxInstrument Interface" on page 170). Further methods proper to

IAdxMapLibor are:

- "IAdxMapLibor Interface: InsertLibor" on page 207
- "IAdxMapLibor Interface: RemoveLibor" on page 207
- "IAdxMapLibor Interface: Init" on page 207

See also

- "AdxMapLibor" on page 269

IAdxMapLibor Interface: InsertLibor

`InsertLibor ()`

Use this method to insert an `AdxLibor` into the stl map.

Arguments

This method takes an `IAdxLibor` pointer as an argument.

IAdxMapLibor Interface: RemoveLibor

`RemoveLibor ()`

Use this method to remove an `AdxLibor` from the stl map.

Arguments

This method takes a BSTR as an argument which is the name of the IAdxLibor to remove.

IAdxMapLibor Interface: Init

`Init ()`

This method must be called to create an `IAdxMapLibor` object.

Arguments

None

IAdxNToDefaultCDS

he `IAdxNToDefaultCDS` interface provides methods and properties to calculate the net present value of an Nth to Default CDS, given its spread, using a copula model specified by the `CreditStructure` argument. The future object properties are listed in the `AdxAttrNToDefaultCDS` enumerated type.

Properties

According to the interface hierarchy, the `IAdxNToDefaultCDS` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via IAdxInstrument.

Methods

The `IAdxNToDefaultCDS` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the IAdxInstrument methods (see "IAdxInstrument Interface" on page 170). Further methods proper to IAdxNToDefaultCDS are:

- "IAdxFuture Interface: Init" on page 201

See also

- "AdxNToDefaultCDS" on page 269
- "AdxAttrNToDefaultCDS" on page 324

IAdxNToDefaultCDS: Init

```
Init (iStructure As String)
```

This method must be called to create an `IAdxNToDefaultCDS` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object

IAdxOpBinary Interface

The `IAdxOpBinary` interface provides methods and properties to manage the binary exotic options. The binary object properties are listed in the `AdxAttrOpBinary` enumerated type.

Properties

According to the interface hierarchy, the `IAdxOpBinary` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via IAdxInstrument and `IAdxOption`.

Methods

The `IAdxOpBinary` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167), the IAdxInstrument (see "IAdxInstrument Interface" on page 170) and IAdxOption methods (see "IAdxOption Interface" on page 209). Further methods proper to are:

- "IAdxOpBinary: Init" on page 208

See also

- "AdxOpBinary" on page 270
- "AdxAttrOpBinary" on page 325

IAdxOpBinary: Init

```
Init (iStructure As String)
```

This method must be called to create an `IAdxOpBinary` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object

IAdxOpLookBack Interface

The `IAdxOpLookBack` interface provides methods and properties to manage the LookBack exotic option. The `OpLookBack` object properties are listed in the `AdxAttrOpLookBack` enumerated type.

Properties

According to the interface hierarchy, the `IAdxOpLookBack` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via IAdxInstrument and `IAdxOption`.

Methods

The `IAdxOpLookBack` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167), the IAdxInstrument (see "IAdxInstrument Interface" on page 170) and IAdxOption methods (see "IAdxOption Interface" on page 209). Further methods proper to abx are:

- "IAdxOpLookBack: Init" on page 209

See also

- "AdxOpLookBack" on page 272
- "AdxAttrOpLookBack" on page 325

IAdxOpLookBack: Init

```
Init (iStructure As String)
```

This method must be called to create an `IAdxOpLookBack` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object

IAdxOption Interface

The `IAdxOption` interface provides methods and properties to manage options. The option object properties are listed in the `AdxAttrOption` enumerated type.

Properties

According to the interface hierarchy, the `IAdxOption` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via `IAdxInstrument`.

Methods

The `IAdxOption` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the `IAdxInstrument` methods (see "IAdxInstrument Interface" on page 170). Further methods proper to IAdxOption are:

- "IAdxOption Interface: AttachCrossCorrelation" on page 209
- "IAdxOption Interface: Init" on page 210

See also

- "AdxOption" on page 275

IAdxOption Interface: AttachCrossCorrelation

```
AttachCrossCorrelation(ipCrossCorrelation As IAdxCorrelation)
```

Use this method to pass the Crosscorrelation matrix to the option object.

Arguments

`ipCrossCorrelation`	Identifies the CrossCorrelation as an `AdxCorrelation` object

IAdxOption Interface: Init

```
Init(iStructure As String)
```

This method must be called to create an `IAdxOption` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object

IAdxParse Interface

The `IAdxParse` interface provides methods to parse string values.

Properties

According to the interface hierarchy, the `IAdxParse` interface inherits the IAdxObject properties, (see "IAdxObject Interface" on page 167).

Methods

The `IAdxParse` interface also inherits the `IAdxObject` methods (see "IAdxObject Interface" on page 167). Further methods proper to `IAdxFloatLeg` are:

"IAdxParse Interface: Init" on page 210

See also

- "AdxParse" on page 278
- "AdxAttrParse" on page 327

IAdxParse Interface: Init

```
Init(iStructure As String)
```

This method must be called to create an `IAdxParse` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object

IAdxRainbow Interface

The `IAdxRainbow` interface provides methods and properties to manage rainbow exotic options. The rainbow object properties are listed in the `AdxAttrRainbow` enumerated type.

Properties

According to the interface hierarchy, the `IAdxRainbow` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via `IAdxInstrument` and `IAdxOption`.

Methods

The `IAdxRainbow` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167), the IAdxInstrument (see "IAdxInstrument Interface" on page 170) and `IAdxOption` methods (see "IAdxOption Interface" on page 209). No further methods are provided.

See also

- "AdxRainbow" on page 279
- "AdxAttrRainbow" on page 328

IAdxRepo Interface

The `IAdxRepo` interface provides methods and properties to manage repurchase agreement. The Repo object properties are listed in the `AdxAttrRepo` enumerated type.

Properties

According to the interface hierarchy, the `IAdxRepo` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via IAdxInstrument.

Methods

The `IAdxRepo` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the IAdxInstrument methods (see "IAdxInstrument Interface" on page 170). Further methods proper to `IAdxRepo` are:

- "IAdxRepo Interface: Init" on page 211

See also

- "AdxRepo" on page 282
- "AdxAttrRepo" on page 330

IAdxRepo Interface: Init

```
Init(iStructure As String)
```

This method must be called to create an `IAdxRepo` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object

IAdxRateModel Interface

The `IAdxRateModel` interface provides methods and properties to manage rate models. The rate model object properties are listed in the `AdxAttrRateModel` enumerated type.

Properties

According to the interface hierarchy, the `IAdxRateModel` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via IAdxModel.

Methods

The `IAdxRateModel` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the IAdxModel (see "IAdxModel Interface" on page 173) methods. Further methods proper to `IAdxRateModel` are:

- "IAdxRateModel Interface: CalcDf" on page 212
- "IAdxRateModel Interface: CalcRate" on page 212
- "IAdxRateModel Interface: Init" on page 212
- "IAdxRateModel Interface: RateConverter" on page 213

See also

- "AdxRateModel" on page 282
- "AdxAttrRateModel" on page 328

IAdxRateModel Interface: CalcDf

```
CalcDf(iStartDate As Variant, iEndDate As Variant) As Variant
```

Use this method to retrieve the equivalent discount factor between StartDate and EndDate.

By definition, the discount factor is the factor by which a future redemption cash flow is multiplied to calculate its present value at StartDate.

Arguments

`iStartDate`	A `Variant` value that specifies the StartDate
`iEndDate`	A `Variant` value that specifies the EndDate

IAdxRateModel Interface: CalcRate

```
CalcRate(iStartDate As Variant, iEndDate As Variant) As Variant
```

Use this method to retrieve the forward rate between StartDate and EndDate.

By definition, a forward rate is an interest rate fixed between two parties for an agreed future lending period.

Arguments

`iStartDate`	A `Variant` value that specifies the StartDate
`iEndDate`	A `Variant` value that specifies the EndDate

IAdxRateModel Interface: Init

```
Init(iStructure As String)
```

This method must be called to create an `IAdxRateModel` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object

IAdxRateModel Interface: RateConverter

`RateConverter(iStartDate, iEndDate, iVarDcb, pFrq As AdxFrq, iVarRateType)` As Double

This method must be called to create an `IAdxRateModel` object.

Arguments

`iStartDate`

`iEndDate`

`iVarDcb`

`pFrq`

`iVarRateType`

IAdxRiskModel Interface

The `IAdxRiskModel` interface provides properties and methods to manage risk models. The risk model object properties are listed in the `AdxAttrRiskModel` enumerated type.

Properties

According to the interface hierarchy, the `IAdxRiskModel` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via `IAdxModel`.

Methods

The `IAdxRiskModel` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the IAdxModel (see "IAdxModel Interface" on page 173) methods. Further methods proper to `IAdxRiskModel` are:

- "IAdxRiskModel Interface: Init" on page 213
- "IAdxRiskModel Interface: CalcDefaultProbability" on page 213

See also

- "AdxRiskModel" on page 285
- "AdxAttrRiskModel" on page 331

IAdxRiskModel Interface: Init

`Init(iStructure As String)`

This method must be called to create an `IAdxRiskModel` object.

Arguments

`iStructure` A `String` describing the values required to initialize the object

IAdxRiskModel Interface: CalcDefaultProbability

`CalcDefaultProbability(iStartDate As Variant, iEndDate As Variant, opProba) As Double`

Use this method to calculate the default probability between StartDate and EndDate. A default probability represents the risk for an instrument to default before maturity.

Arguments

`iStartDate`	A `Variant` value that specifies the StartDate
`iEndDate`	A `Variant` value that specifies the EndDate
`opProba`	A pointer addressed to the default probability

IAdxSchedule

The `IAdxSchedule` object properties are listed in the `AdxAttrSchedule` enumerated types.

Properties

According to the interface hierarchy, the `IAdxSchedule` interface inherits the `IAdxObject` properties, (see "IAdxObject Interface" on page 167) via IAdxInstrument.

Methods

The `IAdxSchedule` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the IAdxInstrument methods (see "IAdxInstrument Interface" on page 170). Further methods proper to IAdxAsian are:

- "IAdxAsian Interface" on page 179

See also

- "AdfinX Analytics Parameters and Constants" on page 295

IAdxSchedule Interface: AttachRateModel

```
AttachRateModel (ildRateModel As Long, ipRateModel As AdxRateModel)
```

This method must be called to attach a rate model.

Arguments

`ildRateModel`

`ipRateModel`

IAdxSchedule Interface: Init

```
Init (iStructure As String)
```

This method must be called to create an `IAdxSchedule` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object

IAdxSchedule Interface: UpdateLineValues

```
UpdateLineValues (ipMrvRange, iLine As Long)
```

This method must be called to update line values.

Arguments

`ipMrvRange`

`iLine`

IAdxStyle Interface

The `IAdxStyle` interface provides methods and properties to manage styles.

Properties

According to the interface hierarchy, the `IAdxStyle` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167).

Methods

The following methods are:

See also

- "AdxStyle" on page 226

IAdxStyle Interface: AddStructure

```
AddStructure(iStructure As BSTR) As Long
```

Use this method to add a structure that defines a style in the `AdxStyle` object.

Arguments

`iStructure`	A `Variant` value that identifies the structure to be added

IAdxStyle Interface: GetCode

```
GetCode() As BSTR
```

Use this method to retrieve the style code.

Arguments

None

IAdxStyle Interface: GetName

```
GetName() As BSTR
```

Use this method to retrieve the style name.

Arguments

None

IAdxStyle Interface: GetStructure

```
GetStructure() As BSTR
```

Use this method to retrieve the structure describing the instrument style.

Arguments

None

IAdxStyle Interface: GetUpdateDate

```
GetUpdateDate() As Long
```

Use this method to retrieve the last date on which a style was updated.

Arguments

None

IAdxStyle Interface: GetUpdateName

```
GetUpdateName() As BSTR
```

Use this method to retrieve the name of the last user who updated the style.

Arguments

None

IAdxStyle Interface: Init

```
Init(iStyleTable As AdxStyleTable, iStructure As String)
```

Use this method to initialize the style used by an object.

Arguments

`iStyleTableName`	A `Variant` value that identifies the name of the style table
`iStructure`	A `String` describing the values required to initialize the object

IAdxStyle Interface: IsPublic

```
IsPublic() As Variant_Boolean
```

Use this method to determine whether the style is public or private.

Arguments

None

IAdxStyle Interface: IsReadOnly

```
IsReadOnly() As Variant_Boolean
```

Use this method to determine whether the style is in read-only mode.

Arguments

None

IAdxStyle Interface: SetCode

```
SetCode(iStyleCode As BSTR) As Long
```

Use this method to set the style code of the style object.

Arguments

None

IAdxStyle Interface: SetName

```
SetName(iStyleName As BSTR) As Long
```

Use this method to set the long style name of the `IAdxStyle` object.

Arguments

None

IAdxStyle Interface: SetReadOnly

```
SetReadOnly(iReadOnly As Variant_Boolean) As Long
```

Use this method to set the read-only mode of the style.

Arguments

None

IAdxStyleTable Interface

The `IAdxStyleTable` interface provides methods and properties to manage the style tables.

Properties

According to the interface hierarchy, the `IAdxStyleTable` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167). Further properties proper to `IAdxStyleTable` are:

- "IAdxStyleTable Interface: Count" on page 218
- "IAdxStyleTable Interface: Item" on page 218

Methods

The following methods are:

See also

IAdxStyleTable Interface: Count

```
Count() As Long
```

This is a read-only property.

Arguments

None

IAdxStyleTable Interface: Item

```
Item(iStyleCode As String)
```

This is a read-only property.

Arguments

This property has no arguments.

IAdxStyleTable Interface: AddStyle

```
AddStyle(pStyle As IAdxStyle) As Long
```

Use this method to add a style to the `AdxStyleTable` object.

Arguments

`pStyle`	Identifies the style to add as an `AdxStyle` object

IAdxStyleTable Interface: DeleteStyle

```
DeleteStyle(iStyleCode As BSTR) As Long
```

Use this method to delete the style from the `AdxStyleTable` object.

Arguments

`iStyleCode`	A `Variant` value that specifies the code to identify the style

IAdxStyleTable Interface: EnumerateStyle

```
EnumerateStyle(pStyle As IAdxStyle) As Variant_Boolean
```

Use this method to enumerate all the styles in the style table. The number of times to call this function must correspond to the number of elements. The function returns the next style in the table. Pass an empty style to the function to begin the enumeration.

Arguments

`pStyle`	Style to start the enumeration

IAdxStyleTable Interface: ExistStyle

```
ExistStyle(iStyleCode As BSTR) As Variant_Boolean
```

Use this method to determine whether the input style exists in the `AdxStyleTable` object.

Arguments

`iStyleCode`	A `Variant` value that specifies the code to identify the style

IAdxStyleTable Interface: GetStyle

```
GetStyle(iStyleCode As BSTR) As IAdxStyle
```

Use this method to retrieve the style contained in the `AdxStyleTable` object.

Arguments

`iStyleCode`	A `Variant` value that specifies the code to identify the style

IAdxStyleTable Interface: GetStyleCodeMaxLen

```
GetStyleCodeMaxLen() As Long
```

Use this method to return the maximum length of a style code in a style table.

Arguments

None

IAdxStyleTable Interface: GetStyleCodeMinLen

```
GetStyleCodeMinLen() As Long
```

Use this method to return the minimum length of a style code in a style table.

Arguments

None

IAdxStyleTable Interface: GetTableName

```
GetTableName() As BSTR
```

Use this method to retrieve the name of a style table.

Arguments

None

IAdxStyleTable Interface: Load

```
Load() As Long
```

Use this method to load a style table.

Arguments

None

IAdxStyleTable Interface: Save

```
Save() As Long
```

Use this method to save current the style table.

Arguments

None

IAdxStyleTable Interface: Init

```
Init(iStyleTable As IAdxStyleTable, iStructure As BSTR)
```

You must call this method to create an `IAdxStyleTable` object.

Arguments

`iStyleTableName`	A `Variant` value that identifies the name of the style table
`iStructure`	A `String` describing the values required to initialize the object

IAdxSwap Interface

The `IAdxSwap` interface provides methods and properties to manage swaps. The swap object properties are listed in the `AdxAttrSwap` enumerated type.

Properties

According to the interface hierarchy, the `IAdxSwap` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via IAdxInstrument.

Methods

The `IAdxSwap` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the IAdxInstrument methods (see "IAdxInstrument Interface" on page 170). Further methods proper to IAdxSwap are provided.

See also

- "AdxSwap" on page 289
- "AdxAttrSwap" on page 334

IAdxSwap Interface: IsPaidLegFixed

```
IsPaidLegFixed() As Variant_Boolean
```

Use this method to determine whether the paid leg is fixed.

Arguments

None

IAdxSwap Interface: IsReceivedLegFixed

```
IsReceivedLegFixed() As Variant_Boolean
```

Use this method to determine whether the received leg is fixed.

Arguments

None

IAdxSwap Interface: Init

```
Init(iStructure As String)
```

This method must be called to create an `IAdxSwap` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object

IAdxTermStructure Interface

The `IAdxTermStructure` interface provides methods used to manage term structures. The term structure object properties are listed in the `AdxAttrTermStructure` enumerated type.

Properties

According to the interface hierarchy, the `IAdxTermStructure` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167).

Methods

The `IAdxTermStructure` interface also inherits the `IAdxObject` methods (see "IAdxObject Interface" on page 167). Further methods proper to IAdxTermStructure are:

- "IAdxTermStructure Interface: AddUnderlying" on page 222
- "IAdxTermStructure Interface: BuildTermStructure" on page 222
- "IAdxTermStructure Interface: Calibrate" on page 222
- "IAdxTermStructure Interface: SetInstrument" on page 222
- "IAdxTermStructure Interface: Init" on page 223

See also

- "IAdxObject Interface" on page 167
- "AdxTermStructure" on page 292
- "AdxAttrTermStructure" on page 337

IAdxTermStructure Interface: AddUnderlying

```
AddUnderlying(ipInstrument As IAdxInstrument) As Variant
```

Use this method to add an underlying instrument used to extend a yield curve built using the `AdxTermStructure` object.

Arguments

`ipInstrument`	Identifies the added instrument as an AdxInstrument object

IAdxTermStructure Interface: BuildTermStructure

```
BuildTermStructure() As Variant
```

Use this method to build a zero coupon yield curve using standard Bootstrapping or the Vasicek Fong model from a set of financial products (Deposits, Bonds, Futures, FRAs, Swaps). Before using the method, you must specify the set of financial products using the `SetInstrumentArray` method.

Arguments

None

IAdxTermStructure Interface: Calibrate

```
Calibrate() As Variant
```

Use this method to perform model calibration using swaptions, caps, and floors. These are called calibrating instruments. The calibration method enables you to estimate the short rate volatility for different maturities.

Arguments

None

IAdxTermStructure Interface: SetInstrument

```
SetInstrumentArray(iInstrumentArray As Variant) As Long
```

Use this method to set an array of instruments used to build the term structure. The set of instruments can be any of the following:

- Deposits
- STIR futures
- Interest Rate Swaps
- Bonds

Each line of the array must contain the following instrument data:

- Type
- Start Date
- Maturity Date

- Coupon
- Market Price & Rate
- Offset (optional)
- Structure or style

Arguments

`iInstrumentArray`	A `Variant` value that identifies the array of instruments

IAdxTermStructure Interface: Init

```
Init(iStructure As String)
```

This method must be called to create an `IAdxTermStructure` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object

IAdxTimeSeries Interface

The `IAdxTimeSeries` interface allows you to calculate regression parameters and ratios in a Time Series.

Properties

According to the interface hierarchy, the `IAdxTimeSeries` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167).

Methods

The `IAdxTimeSeries` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the IAdxInstrument methods (see "IAdxInstrument Interface" on page 170). Further methods proper to IAdxTimeSeries are:

"IAdxTimeSeries Interface: Init" on page 223

See also

"AdxTimeSeries" on page 294

IAdxTimeSeries Interface: Init

```
Init(iTS As String)
```

Use this method to initialize your `TimeSeries` object using a structure of appropriate keywords.

Arguments

`iTS`	A `String` containing the appropriate keywords used to initialize the object.

IAdxVolatilityModel Interface

The `IAdxVolatilityModel` interface provides methods and properties to manage volatility models. The volatility model object properties are listed in the `AdxAttrVolatilityModel` enumerated type.

Properties

According to the interface hierarchy, the `IAdxVolatilityModel` interface inherits the `IAdxObject` properties (see "IAdxObject Interface" on page 167) via IAdxModel.

Methods

The `IAdxVolatilityModel` interface also inherits the `IAdxObject` (see "IAdxObject Interface" on page 167) and the `IAdxModel` (see "IAdxModel Interface" on page 173) methods. Further method proper to `IAdxVolatilityModel` is:

- "IAdxVolatilityModel Interface: Init" on page 224

See also

- "AdxVolatilityModel" on page 294
- "AdxAttrVolatilityModel" on page 341

IAdxVolatilityModel Interface: Init

```
Init (iStructure As String)
```

This method must be called to create an `IAdxVolatilityModel` object.

Arguments

`iStructure`	A `String` describing the values required to initialize the object

AdfinX Analytics Objects Utilities

AdxCalendar

The AdxCalendar component object implements the IAdxCalendar interface.

VBA sample

```
'This example shows how to create a calendar and perform calculations:
'Control declaration and initialization
  Dim Calendar As AdxCalendar
  Set Calendar = new AdxCalendar
'Variable declaration
  Dim CalendarStructure As Variant
  Dim CalcDate As Variant
  Dim ResultDate As Variant
'Set Calendar attributes and parameters
  Calendar.ErrorMode = DialogBox
  CalendarStructure = "UKG"
  CalcDate = "01MAR04"
  Calendar.Init CalendarStructure
'Perform a calculation
  ResultDate = Calendar.AddMonths 3, AdxEndOfMonthConvention.ADX_EMC_LAST,
AdxDateMovingConvention.ADX_DMC_NO
```

See also

"IAdxIdxStyle Interface" on page 202

AdxIdxStyle

The AdxIdxStyle component object implements the IAdxIdxStyle interface.

VBA sample

```
'This example shows how to create an idx style:
'Control declaration and initialization
  Dim IdxStyle As AdxIdxStyle
  Set IdxStyle = new AdxIdxStyle
'Variable declaration
Dim IdxStyleCode As Variant
'Set Style attributes and parameters
  IdxStyle.ErrorMode = DialogBox
  IdxStyleCode = "AGBCPI"
  IdxStyle.LoadHistory IdxStyleCode
```

See also

"IAdxIdxStyle Interface" on page 202

AdxStyle

The AdxStyle component object implements the IAdxStyle interface.

VBA sample

```
'This example shows how to create a style:
'Control declaration and initialization
  Dim Style As AdxStyle
  Set Style = new AdxStyle
  Dim StyleTable As AdxStyleTable
  Set StyleTable = new AdxStyleTable
'Variable declaration
  Dim StyleName As Variant
  Dim StyleTableName As Variant
'Set Style attributes and parameters
  StyleTable.ErrorMode = DialogBox
  StyleTableName = "BOND"
  StyleTable.Init = StyleTableName
  Style.ErrorMode = DialogBox
  StyleName = "OAT"
  Style.InitStyle StyleTable, StyleName
```

See also

"IAdxIdxStyle Interface" on page 202

AdxStyleTable

The AdxStyleTable component object implements the IAdxStyleTable interface.

VBA sample

```
'This example shows how to create a style table:
'Control declaration and initialization
  Dim StyleTable As AdxStyleTable
  Set StyleTable = new AdxStyleTable
'Variable declaration
  Dim StyleTableName As Variant
'Set Style table attributes and parameters
  StyleTable.ErrorMode = DialogBox
  StyleTableName = "FOREX"
  StyleTable.Init = StyleTableName
```

See also

"IAdxStyleTable Interface" on page 217

AdfinX Analytics Objects

AdfinX Analytics Object Overview

AdfinX Analytics 3.0 Component Library implements the interfaces that are derived from the IDispatch interface. They enable access by Active X clients, such as Visual Basic. They inherit the properties and methods of these interfaces.

The following data object components are provided:

- The controls to manage financial products including bonds, FRNs, options, etc.
- The controls to manage different models including interest rate models, volatility models, and dividend models.
- The controls to use different numerical procedure including calculations based on formulas, trees, or finite difference.

ⓘ Code samples are given when using ActiveX controls from the Visual Basic IDE.

AdxAlgorithm

The `AdxAlgorithm` component object implements the `IAdxAlgorithm` interface which inherits its properties and methods from "IAdxObject Interface" on page 167.

See also

"IAdxAlgorithm Interface" on page 176

AdxAsian

The `AdxAsian` component object implements the `IAdxAsian` interface that inherits its properties and methods from `IAdxInstrument`.

The payoff of an Asian option is based on the average level of the underlying asset price over a period of time, rather than only on its final value.

VBA sample

This example shows how to create an Asian asset. Once this has been defined you can use the sample defined in AdxOption with the Asian instrument as the underlying of the option to compute the premium of an Asian option.

```
''Control declaration and initialization
  Dim Asian As AdxAsian
  Set Asian = new AdxAsian
  Dim Asset As AdxAsset
  Set Asset = new AdxAsset
'Variable declaration
  Dim AsianStructure As Variant
  Dim AssetStructure As Variant
  Dim FirstFixingDate As Variant
  Dim SpotPrice As Variant
  Dim AvgPrice As Variant
  Dim NbFixing As Variant
  Dim Volatility As Variant
'Definitions of option attributes to calculate
  Dim Premium As Variant
'Set Asset attributes and parameters
  SpotPrice = 1.82
  AssetStructure = "UI:FUT"
  Asset.Init AssetStructure
  Asset.SetAttribute AdxAttrAsset.ADX_ATTR1F_ASSET_PRICE, SpotPrice
'Set VolatilityModel attributes
  VolatilityModel.ErrorMode = DialogBox
  Volatility = 0.18
  VolatilityModel.SetAttribute AdxAttrVolatilityModel.ADX_ATTR1R_VOLATILITY,
Volatility
```

```
'Attach the Volatility to the Asset object
 Asset.AttachModel AdxAttrAsset.ADX_PTR_MODEL_DIVIDEND, VolatilityModel
'Set option attributes and parameters
 Asian.ErrorMode = DialogBox
 AsianStructure = "AVE:ARI ASIAN:RATE"
 FirstFixingDate = "10JAN01"
 ExpiryDate = "01JUL01"
 AvgPrice = 1.75
 NbFixing = 10
 Asian.Init AsianStructure
 Asian.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_SETTLEMENT, CalcDate
 Asian.SetAttribute AdxAttrAsian.ADX_ATTR1D_FIRST_FIXING, FirstFixingDate
 Asian.SetAttribute AdxAttrAsian.ADX_ ATTR1F_AVERAGE, AvgPrice
```

See also

- "IAdxAsian Interface" on page 179
- "AdxAttrAsian" on page 297

AdxAsset

The `AdxAsset` component object implements the `IAdxAsset` interface that inherits its properties and methods from `IAdxInstrument`. The asset instruments can be indexes, commodities, and stocks. Indexes and stocks can generate cashflows during their life; commodities imply storage and insurance costs.

VBA sample

See "AdxOption" on page 275 to see how to use `AdxAsset`.

See also

- "IAdxAsset Interface" on page 180
- "AdxAttrAsset" on page 297

AdxAssetSwap

The `AdxAssetSwap` component object implements the `IAdxAssetSwap` interface which inherits its properties and methods from "IAdxObject Interface" on page 167.

See also

"IAdxAssetSwap Interface" on page 180

AdxBarrierCapFloor

The `AdxBarrierCapFloor` component object implements the `IAdxBarrierCapFloor` interface that inherits its properties and methods from `IAdxCapFloor` and `IAdxFloatLeg`. Barrier caps and floors are interest rate risk management products based on barrier options.

VBA sample

```
'Control declaration and initialization
 Dim BarrierCapFloor As AdxBarrierCapFloor
 Set BarrierCapFloor = new AdxBarrierCapFloor
 Dim RateModel As AdxRateModel
 Set RateModel = new AdxRateModel
 Dim CalcMethod As AdxCalcMethod
 Set CalcMethod = new AdxCalcMethod
'Variable declaration
 Dim Structure As String
 Dim CapFloorStructure As String
 Dim RateStructure As String
 Dim CalcStructure As String
 Dim CalcDate As Variant
 Dim StartDate As Variant
 Dim Maturity As Variant
 Dim FirstRate As Double
 Dim Strike As Double
 Dim BarrierUp As Variant
 Dim BarrierDown As Variant
 Dim RebateUp As Variant
 Dim RebateDown As Variant
'Definitions of BarrierCapFloor attributes to calculate
 Dim Premium As Variant
```

```
 BarrierCapFloor.SetAttribute  AdxAttrBarrierCapFloor.ADX_ATTR1R_CAPFLOOR_
BARRIER_DOWN, BarrierDown
 BarrierCapFloor.SetAttribute  AdxAttrBarrierCapFloor.ADX_ATTR1R_CAPFLOOR_
REBATE_DOWN, RebateDown
‘Set RateModel attributes and parameters
 RateModel.ErrorMode = DialogBox
 RateStructure = "RM:BS ZCTYPE:RATE DCB:A5 RATEFRQ:ZERO IM:LIN RATETYPE:ACT
CLDRADJ:CLDR"
 StartDate = "25NOV02"
 Dim RateArray as Variant
 ReDim Preserve RateArray (0 To 9, 0 To 1)
RateArray (0, 0) = "13NOV02"
 RateArray (1, 0) = "16NOV02"
 RateArray (2, 0) = "20DEC02"
 RateArray (3, 0) = "11JAN03"
 RateArray (4, 0) = "12DEC03"
 RateArray (5, 0) = "17JAN05"
 RateArray (6, 0) = "17MAY05"
 RateArray (7, 0) = "15DEC05"
 RateArray (8, 0) = "17JAN07"
 RateArray (9, 0) = "15MAY07"
 RateArray (0, 1) = 0.0337
 RateArray (1, 1) = 0.03375
 RateArray (2, 1) = 0.032
 RateArray (3, 1) = 0.033756
 RateArray (4, 1) = 0.0331
 RateArray (5, 1) = 0.02
 RateArray (6, 1) = 0.02
```

```
RateArray (7, 1) = 0.02
 RateArray (8, 1) = 0.04
 RateArray (9, 1) = 0.041
 RateModel.Init (RateStructure)
 RateModel.SetAttribute AdxAttrRateModel.ADX_ATTR1R_RATEMODEL_ARRAY,
RateArray
'Attach the RateModel to the CapFloor object
 BarrierCapFloor.AttachModel ADX_PTR_MODEL_DISCOUNTRATE, RateModel
‘Calculation initialization
'Set calculation formula
 CalcMethod.ErrorMode = DialogBox
 CalcStructure = "CMT:FORM FT:BS"
 CalcMethod.Init (CalcStructure)
'Ask the formula pricer to compute the calculation attributes of your barriercapfloor
'Instrument passed to the calculation method
'The AskToCompute method is called for each attribute to calculate
 BarrierCapFloor.AskToCompute(AdxAttrCapFloor.ADX_ATTR3F_CAPFLOOR_PREMIUM)
 CalcMethod.AttachInstrument BarrierCapFloor
'Level #3 attributes computation
 CalcMethod.Compute
'Get the computed value
 Premium =  BarrierCapFloor.GetAttribute(AdxAttrCapFloor.ADX_ATTR3F_CAPFLOOR_
PREMIUM)
```

See also

- "IAdxBarrierCapFloor Interface" on page 181
- "AdxAttrBarrierCapFloor" on page 299

AdxBasket

The `AdxBasket` component object implements the `IAdxBasket` interface that inherits its properties and methods from `IAdxInstrument`.

Basket options, also called portfolio options, are a variation of rainbow options. Their payoff is the weighted average of the prices within a basket of underlying assets.

VBA sample

```
'This example shows how to create a basket of two assets. Once this has been defined you can use the sample defined in AdxOption with the basket instrument as the underlying of the option to compute the premium of a basket option:
'Control declaration and initialization
 Dim Basket As AdxBasket
 Set Basket = new AdxBasket
 Dim Asset1 As AdxAsset
 Set Asset1 = new AdxAsset
 Dim Asset2 As AdxAsset
 Set Asset2 = new AdxAsset
 Dim Correlation As AdxCorrelation
 Set Correlation = new AdxCorrelation
'Variable declaration
 Dim Asset1Structure As Variant
 Dim Asset2Structure As Variant
 Dim SpotPrice1 As Variant
 Dim SpotPrice2 As Variant
 Dim CorrelationArray As Variant
 Dim WeightArray As Variant
'Set Basket attributes and parameters
 Basket.ErrorMode = DialogBox
 CalcDate = "01MAR04"
 Redim Preserve CorrelationArray (0 To 1, 0 To 1)
 CorrelationArray (0,0) = 0.19
 CorrelationArray (0,1) = -0.5
 CorrelationArray (1,0) = -0.5
 CorrelationArray (1,1) = 0.0725
```

```
 Correlation.SetAttribute AdxAttrCorrelation.ADX_ATTR1R_COR_MATRIX,
CorrelationArray
 Redim Preserve WeightArray (0 To 1)
 WeightArray (0) = 0.5
 WeightArray (0) = 0.5
 Basket.Init
 Basket.SetCorrelation CorrelationArray
 Basket.SetWeight WeightArray
'Set Assets attributes and parameters
 SpotPrice1= 73.21
 Asset1Structure = "UI:SEC"
 Asset1.Init(Asset1Structure)
 Asset1.SetAttribute AdxAttrAsset.ADX_ATTR1F_ASSET_PRICE, SpotPrice1
 SpotPrice2= 76.09
 Asset2Structure = "UI:SEC"
 Asset2.Init(Asset2Structure)
 Asset2.SetAttribute AdxAttrAsset.ADX_ATTR1F_ASSET_PRICE, SpotPrice2
'Set assets in the basket
 Basket.AddInstrument Asset1
 Basket.AddInstrument Asset2
```

See also

- "IAdxBasket Interface" on page 182
- "AdxAttrBasket" on page 299

AdxBond

The `AdxBond` component object implements the `IAdxBond` interface that inherits its properties and methods from `IAdxFixedLeg`.

Bonds are instruments that distribute fixed coupons at regular intervals and redemption on a specific date or dates.

VBA sample

```
'Control declaration and initialization
  Dim Bond As AdxBond
  Set Bond = new AdxBond
  Dim RateModel As AdxRateModel
  Set RateModel = new AdxRateModel
  Dim CalcMethod As AdxCalcMethod
  Set CalcMethod = new AdxCalcMethod
'Variable declaration
  Dim BondStructure As Variant
  Dim RateStructure As Variant
  Dim CalcStructure As Variant
  Dim CalcDate As Variant
  Dim MaturityDate As Variant
  Dim Coupon As Variant
  Dim IssueDate As Variant
  Dim Yield As Variant
'Definitions of bond attributes to calculate
  Dim Price As Variant
  Dim Duration As Variant
  Dim AvgLife As Variant
  Dim Convexity As Variant
  Dim YtwytBdDate As Variant
  Dim BondSpread As Variant
  Dim OptionFreePrice As Variant
  Dim GrossPrice As Variant
  Dim CleanPrice As Variant
  Dim PVBP As Variant
```

```
‘Set Bond attributes and parameters
 Bond.ErrorMode = DialogBox
 BondStructure = "FRQ:1 ACC:AA"
 CalcDate = "10JAN01"
 MaturityDate = "25JUL10"
 IssueDate = "25MAR00"
 Coupon = 0.05
 Bond.Init (BondStructure)
 Bond.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_SETTLEMENT, CalcDate
 Bond.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_ISSUE, IssueDate
 Bond.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_MATURITY, MaturityDate
 Bond.SetAttribute AdxAttrFixedLeg.ADX_ATTR2F_FIXEDLEG_COUPON, Coupon
‘Set RateModel attributes and parameters
 RateModel.ErrorMode = DialogBox
 RateStructure = "RM:YTM RATETYPE:CMP DCB:AA RATEFRQ:1"
 YcStartDate = "10OCT00"
 Yield = 0.0475
 RateModel.Init (RateStructure)
 RateModel.SetAttribute AdxAttrRateModel.ADX_ATTR1R_RATEMODEL_ARRAY, Yield
'Attach the RateModel to the Bond object
 Bond.AttachModel ADX_PTR_MODEL_DISCOUNTRATE, RateModel
'Calculation initialization
'Set calculation formula
 CalcMethod.ErrorMode = DialogBox
 CalcStructure = "CMT:FORM"
 CalcMethod.Init (CalcStructure)
'Ask the formula pricer to compute the calculation attributes of your bond
'Instrument passed to the calculation method
'The AskToCompute method is called for each attribute to calculate
 Bond.AskToCompute (AdxAttrBond.ADX_ATTR3F_BOND_PRICE)
```

```
Bond.AskToCompute (AdxAttrBond.ADX_ATTR3F_BOND_DURATION)
 Bond.AskToCompute (AdxAttrBond.ADX_ATTR3F_BOND_YIELD)
 Bond.AskToCompute (AdxAttrBond.ADX_ATTR3F_BOND_AVGLIFE)
 Bond.AskToCompute (AdxAttrBond.ADX_ATTR3D_BOND_YTWYTBDATE)
 Bond.AskToCompute (AdxAttrBond.ADX_ATTR3F_BOND_CONVEXITY)
 Bond.AskToCompute (AdxAttrBond.ADX_ATTR3F_BOND_OPTIONFREEPRICE)
 Bond.AskToCompute (AdxAttrBond.ADX_ATTR3F_BOND_PVBP)
 Bond.AskToCompute (AdxAttrBond.ADX_ATTR3F_BOND_VOLATILITY)
 Bond.AskToCompute (AdxAttrBond.ADX_ATTR3F_BOND_PVBP)
 Bond.AskToCompute (AdxAttrBond.ADX_ATTR3F_BOND_CLEANPRICE)
 Bond.AskToCompute (AdxAttrBond.ADX_ATTR3F_BOND_SPREAD)
 Bond.AskToCompute (AdxAttrBond.ADX_ATTR3F_BOND_GROSSPRICE)
 CalcMethod.AttachInstrument Bond
'Level #3 attributes computation
 CalcMethod.Compute
'Get the computed value
 Accrued = Bond.GetAttribute(AdxAttrLeg.ADX_ATTR2F_LEG_ACCRUED)
 Price = Bond.GetAttribute(AdxAttrBond.ADX_ATTR3F_BOND_PRICE)
 Duration = Bond.GetAttribute(AdxAttrBond.ADX_ATTR3F_BOND_DURATION)
 AvgLife = Bond.GetAttribute(AdxAttrBond.ADX_ATTR3F_BOND_AVGLIFE)
 Duration = Bond.GetAttribute(AdxAttrBond.ADX_ATTR3F_BOND_DURATION)
 Convexity = Bond.GetAttribute(AdxAttrBond.ADX_ATTR3F_BOND_CONVEXITY)
 YtwytBdDate = Bond.GetAttribute(AdxAttrBond.ADX_ATTR3D_BOND_YTWYTBDATE)
 BondSpread = Bond.GetAttribute(AdxAttrBond.ADX_ATTR3F_BOND_SPREAD)
 OptionFreePrice = Bond.GetAttribute(AdxAttrBond.ADX_ATTR3F_BOND_
OPTIONFREEPRICE)
 GrossPrice = Bond.GetAttribute(AdxAttrBond.ADX_ATTR3F_BOND_GROSSPRICE)
 CleanPrice = Bond.GetAttribute(AdxAttrBond.ADX_ATTR3F_BOND_CLEANPRICE)
 PVBP = Bond.GetAttribute(AdxAttrBond.ADX_ATTR3F_BOND_PVBP)
```

See also

- "IAdxBond Interface" on page 183
- "AdxAttrBond" on page 300

AdxCalcMethod

The `AdxCalcMethod` component object implements the `IAdxCalcMethod` interface that inherits its properties and methods from `IAdxObject`.

Calculation methods used to value or price financial products can be an analytical formula, a tree-based method, or a finite-differences formula.

VBA sample

See "AdxBond" on page 235 to see how to use `AdxCalcMethod`.

See also

- "AdxAttrCalcMethod" on page 301
- "AdxAttrFiniteDiff" on page 310
- "AdxAttrFormula" on page 314
- "AdxAttrFxFormula" on page 319
- "AdxAttrTree" on page 340
- "IAdxCalcMethod Interface" on page 172

AdxCapFloor

The `AdxCapFloor` component object implements the `IAdxCapFloor` interface that inherits its properties and methods from `IAdxFloatLeg`.

Caps and floors are interest rate risk management products based on strips of options. Like interest rate swaps, they allow you to hedge a long term exposure that spans multiple periods, each of which succeeds the other, but contrary to IRSs, they still allow you to benefit from advantageous market conditions.

VBA sample

```
'Control declaration and initialization
  Dim CapFloor As AdxCapFloor
  Set CapFloor = new AdxCapFloor
  Dim RateModel As AdxRateModel
  Set RateModel = new AdxRateModel
  Dim CalcMethod As AdxCalcMethod
  Set CalcMethod = new AdxCalcMethod
'Variable declaration
  Dim CapFloorStructure As Variant
  Dim RateStructure As Variant
  Dim CalcStructure As Variant
  Dim CalcDate As Variant
  Dim MaturityString As Variant
  Dim IssueDate As Variant
  Dim CapStrike As Variant
  Dim FloorStrike As Variant
  Dim FirstRate As Variant
'Definitions of CapFloor attributes to calculate
  Dim Premium As Variant
```

```
'Set CapFloor attributes and parameters
 CapFloor.ErrorMode = DialogBox
 CapFloorStructure = "CAP FRQ:2 CCM:MMA5"
 CalcDate = "22JUN99"
 MaturityString = "1Y"
 IssueDate = "22JUN99"
 FirstRate = 0.03
 CapStrike = 0.03295
 FloorStrike = 0
 CapFloor.Init(CapFloorStructure)
 CapFloor.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_SETTLEMENT, CalcDate
 CapFloor.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_ISSUE, IssueDate
 CapFloor.SetAttribute AdxAttrLeg.ADX_ATTR1S_LEG_MATURITY_CODE, MaturityDate
 CapFloor.SetAttribute AdxAttrFloatLeg.ADX_ATTR1F_FLOATLEG_CAP, CapStrike
 CapFloor.SetAttribute AdxAttrFloatLeg.ADX_ATTR1F_FLOATLEG_FLOOR, FloorStrike
 CapFloor.SetAttribute AdxAttrFloatLeg.ADX_ATTR1F_FLOATLEG_CURRENT_INDEX,
FirstRate
'Set RateModel attributes and parameters
 RateModel.ErrorMode = DialogBox
 RateStructure = "RM:BS"
 YcStartDate = "22JUN99"
 Dim RateArray as Variant
 ReDim Preserve RateArray (0 To 2, 0 To 5)
 RateArray (0, 0) = "22JUN99"
 RateArray (1, 0) = "22JUN00"
 RateArray (2, 0) = "22JUN01"
 RateArray (3, 0) = "22JUN02"
 RateArray (4, 0) = "22JUN03"
 RateArray (0, 1) = 0.03
 RateArray (1, 1) = 0.035
```

```
 RateArray (2, 1) = 0.04
 RateArray (3, 1) = 0.045
 RateArray (4, 1) = 0.05
 RateArray (0, 2) = 0.2
 RateArray (1, 2) = 0.2
 RateArray (2, 2) = 0.2
 RateArray (3, 2) = 0.2
 RateArray (4, 2) = 0.2
 RateModel.Init (RateStructure)
 RateModel.SetAttribute AdxAttrRateModel.ADX_ATTR1R_RATEMODEL_ARRAY,
RateArray
'Attach the RateModel to the CapFloor object
CapFloor.AttachModel ADX_PTR_MODEL_DISCOUNTRATE, RateModel
‘Calculation initialization
'Set calculation formula
 CalcMethod.ErrorMode = DialogBox
 CalcStructure = "CMT:FORM"
 CalcMethod.Init (CalcStructure)
'Ask the formula pricer to compute the calculation attributes of your capfloor
'Instrument passed to the calculation method
'The AskToCompute method is called for each attribute to calculate
 CapFloor.AskToCompute(AdxAttrCapFloor.ADX_ATTR3F_CAPFLOOR_PREMIUM)
 CalcMethod.AttachInstrument CapFloor
'Level #3 attributes computation
 CalcMethod.Compute
'Get the computed value
 Premium = CapFloor.GetAttribute(AdxAttrCapFloor.ADX_ATTR3F_CAPFLOOR_PREMIUM)
```

See also

- "IAdxCapFloor Interface" on page 187
- "AdxAttrCapFloor" on page 302

AdxCashFlow

The `AdxCashFlow` component object implements the `IAdxCashFlow` interface that inherits its properties and methods from `IAdxInstrument`.

This object gives you access to customizable cashflow-based products. These products have all the main characteristics of a financial instrument (settlement date, rate model to discount cashflows).

VBA sample

```
'Control declaration and initialization
 Dim CashFlow As Adx CashFlow
 Set CashFlow = new Adx CashFlow
 Dim RateModel As AdxRateModel
 Set RateModel = new AdxRateModel
 Dim CalcMethod As AdxCalcMethod
 Set CalcMethod = new AdxCalcMethod
'Variable declaration
 Dim RateStructure As Variant
 Dim CfRange(1 to 2, 1 to 5) As Variant
 Dim CalcStructure As Variant
 Dim CalcDate As Variant
 Dim MaturityDate As Variant
 Dim Coupon As Variant
 Dim IssueDate As Variant
 Dim Yield As Variant
'Definitions of bond attributes to calculate
 Dim Price As Variant
```

```
'Set Bond attributes and parameters
 CashFlow.ErrorMode = DialogBox
 CfRange(1,1) = "17FEB01"
 CfRange(1,2) = "17FEB02"
 CfRange(1,3) = "17FEB03"
 CfRange(1,4) = "17FEB04"
 CfRange(1,5) = "17FEB05"
 CfRange(2,1) = 0.08
CfRange(2,2) = 0.06
 CfRange(2,3) = 0.06
 CfRange(2,4) = 0.06
 CfRange(2,5) = 1.06
 CalcDate = "10JAN01"
 Coupon = 0.05
 CashFlow.Init (CfRange)
 CashFlow.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_SETTLEMENT, CalcDate
 CashFlow.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_SETTLEMENT, CalcDate
'Set RateModel attributes and parameters
 RateModel.ErrorMode = DialogBox
 RateStructure = "RM:YTM RATETYPE:CMP DCB:AA RATEFRQ:1"
 YcStartDate = "10OCT00"
 Yield = 0.0475
 RateModel.Init (RateStructure)
 RateModel.SetAttribute AdxAttrRateModel.ADX_ATTR1R_RATEMODEL_ARRAY, Yield
'Attach the RateModel to the Bond object
 CashFlow.AttachModel ADX_PTR_MODEL_DISCOUNTRATE, RateModel
'Calculation initialization
'Set calculation formula
 CalcMethod.ErrorMode = DialogBox
 CalcStructure = "CMT:FORM"
 CalcMethod.Init (CalcStructure)
'Ask the formula pricer to compute the calculation attributes of your bond
'Instrument passed to the calculation method
'The AskToCompute method is called for each attribute to calculate
 CashFlow.AskToCompute (AdxAttrCashFlow.ADX_ATTR3F_CASHFLOW_PRICE)
CalcMethod.AttachInstrument CashFlow
Level #3 attributes computation
 CalcMethod.Compute
'Get the computed value
 Price = CashFlow.GetAttribute(AdxAttrCashFlow. ADX_ATTR3F_CASHFLOW_PRICE)
```

See also

- "IAdxCashFlow Interface" on page 188
- "AdxAttrCashFlow" on page 303

AdxCDOTranche

The `AdxCDOTranche` component object implements the `AdxCDOTranche` interface tthat inherits its properties and methods from `IAdxInstrument`. This object is used to calculate the spread of a CDO tranche paid by the protection buyer, using a copula model specified by the `CreditStructure` argument.

See also

- "AdfinX Analytics Interfaces" on page 167
- "AdfinX Analytics Interfaces" on page 167
- "AdfinX Analytics Parameters and Constants" on page 295

AdxChooser

The `AdxChooser` component object implements the `IAdxChooser` interface that inherits its properties and methods from `IAdxInstrument`.

Chooser options allow the holder to choose at some pre-determined future date whether the option is a call or a put, with the same predefined strike price and expiry date. Chooser options are cheaper than straddles because the holder cannot benefit from both the call option and the put option until maturity.

VBA sample

```
'Control declaration and initialization
Dim Chooser As AdxChooser
 Set Chooser = new AdxChooser
 Dim Asset As AdxAsset
 Set Asset = new AdxAsset
 Dim RateModel As AdxRateModel
 Set RateModel = new AdxRateModel
 Dim VolatilityModel As AdxVolatilityModel
 Set VolatilityModel = new AdxVolatilityModel
 Dim CalcMethod As AdxCalcMethod
 Set CalcMethod = new AdxCalcMethod
'Variable declaration
 Dim OptionStructure As Variant
 Dim UnderlyingStructure As Variant
 Dim RateStructure As Variant
 Dim CalcStructure As Variant
 Dim CalcDate As Variant
 Dim SpotPrice As Variant
 Dim Volatility As Variant
 Dim RiskFreeRate As Variant
 Dim ExpiryDate As Variant
 Dim StrikePrice As Variant
'
```

```
'Definitions of option attributes to calculate
 Dim Premium As Variant'
'Set Asset attributes and parameters
 SpotPrice = 134
 Asset.SetAttribute AdxAttrAsset.ADX_ATTR1F_ASSET_PRICE, SpotPrice
'Set VolatilityModel attributes
 VolatilityModel.ErrorMode = DialogBox
 Volatility = 0.18
 VolatilityModel.SetAttribute  AdxAttrVolatilityModel.ADX_ATTR1R_
VOLATILITY,Volatility
'Attach the Volatility to the Asset object
 Asset.AttachModel ADX_PTR_MODEL_DIVIDEND, VolatilityModel
'Set option attributes and parameters
 Chooser.ErrorMode = DialogBox
 OptionStructure = "CALL EXM:A"
 CalcDate = "10JAN01"
 ExpiryDate = "25JUL02"
 StrikePrice = 136
 Chooser.Init (OptionStructure)
 Chooser.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_SETTLEMENT, CalcDate
 Chooser.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_MATURITY, ExpiryDate
 Chooser.SetAttribute AdxAttrOption.ADX_ATTR1F_OPT_STRIKE, StrikePrice
'Set RateModel attributes and parameters
 RateModel.ErrorMode = DialogBox
 RateStructure = "RM:YTM RATETYPE:CONT"
 RiskFreeRate = 0.0475
 RateModel.Init (RateStructure)
 RateModel.SetAttribute AdxAttrRateModel.ADX_ATTR1R_RATEMODEL_ARRAY,
RiskFreeRate
```

```
'Attach the RateModel to the Option object
 Chooser.AttachModel ADX_PTR_MODEL_DISCOUNTRATE, RateModel
 Chooser.AttachInstrument ADX_PTR_INSTRUMENT_UNDERLYING1, Asset
'Calculation initialization
'Set calculation formula
 CalcMethod.ErrorMode = DialogBox
 CalcStructure = "CMT:FORM"
 CalcMethod.Init (CalcStructure)
'Ask the formula pricer to compute the calculation attributes of your bond
'Instrument passed to the calculation method
'The AskToCompute method is called for each attribute to calculate
 Chooser.AskToCompute (AdxAttrOption.ADX_ATTR3F_OPT_PREMIUM)
 CalcMethod.AttachInstrument Option
'Level #3 attributes computation
 CalcMethod.Compute
'Get the computed value
 Premium = Chooser.GetAttribute(AdxAttrOption.ADX_ATTR3F_OPT_PREMIUM)
```

See also

"IAdxChooser Interface" on page 189

AdxConvBond

The `AdxConvBond` component object implements the `IAdxConvBond` interface that inherits its properties and methods from `IAdxFixedLeg`.

Convertible bonds are securities that give the holder the option to convert into another security at predefined dates and rates.

VBA sample

```
'Control declaration and initialization
 Dim ConvBond As AdxConvBond
 Set ConvBond = new AdxConvBond
 Dim Asset As AdxAsset
 Set Asset = new AdxAsset
 Dim VolatilityModel As AdxVolatilityModel
 Set VolatilityModel = new AdxVolatilityModel
 Dim RateModel As AdxRateModel
 Set RateModel = new AdxRateModel
 Dim CalcMethod As AdxCalcMethod
 Set CalcMethod = new AdxCalcMethod
'Variable declaration
 Dim ConvBondStructure As Variant
 Dim RateStructure As Variant
 Dim CalcStructure As Variant
 Dim UnderlyingStructure As Variant
 Dim SpotPrice As Variant
 Dim Volatility As Variant
 Dim CalcDate As Variant
 Dim MaturityDate As Variant
 Dim Coupon As Variant
 Dim IssueDate As Variant
'Definitions of bond attributes to calculate
 Dim Price As Variant
```

```
'Set Asset attributes and parameters
 SpotPrice = 134
 UnderlyingStructure = "UI:SEC"
 Asset.Init(UnderlyingStructure)
 Asset.SetAttribute AdxAttrAsset.ADX_ATTR1F_ASSET_PRICE, SpotPrice
'Set VolatilityModel attributes
 VolatilityModel.ErrorMode = DialogBox
 Volatility = 0.18
 VolatilityModel.SetAttribute  AdxAttrVolatilityModel.ADX_ATTR1R_
VOLATILITY,Volatility
'Attach the Volatility to the Asset object
 Asset.AttachModel ADX_PTR_MODEL_DIVIDEND, VolatilityModel
'Set Bond attributes and parameters
 ConvBond.ErrorMode = DialogBox
 ConvBondStructure = "FRQ:1 ACC:AA"
 CalcDate = "10JAN01"
 MaturityDate = "25JUL10"
 IssueDate = "25MAR00"
 Coupon = 0.05
 ConvBond.Init (BondStructure)
 ConvBond.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_SETTLEMENT, CalcDate
 ConvBond.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_ISSUE, IssueDate
 ConvBond.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_MATURITY,
MaturityDate
 ConvBond.SetAttribute AdxAttrFixedLeg.ADX_ATTR2F_FIXEDLEG_COUPON, Coupon
'Set RateModel attributes and parameters
 RateModel.ErrorMode = DialogBox
 RateStructure = "RM:YTM RATETYPE:CMP DCB:AA RATEFRQ:1"
 YcStartDate = "10OCT00"
 Yield = 0.0475
 RateModel.Init (RateStructure)
 RateModel.SetAttribute AdxAttrRateModel.ADX_ATTR1R_RATEMODEL_ARRAY, Yield
```

```
Attach the RateModel to the Bond object
 ConvBond.AttachModel ADX_PTR_MODEL_DISCOUNTRATE, RateModel'
Calculation initialization
‘Set calculation formula
 CalcMethod.ErrorMode = DialogBox
 CalcStructure = "CMT:TREE"
 CalcMethod.Init (CalcStructure)
'Ask the formula pricer to compute the calculation attributes of your bond
'Instrument passed to the calculation method
'The AskToCompute method is called for each attribute to calculate
 ConvBond.AskToCompute (AdxAttrBond.ADX_ATTR3F_BOND_PRICE)
 CalcMethod.AttachInstrument ConvBond
'Level #3 attributes computation
 CalcMethod.Compute
'Get the computed value
 Price = ConvBond.GetAttribute(AdxAttrBond.ADX_ATTR3F_BOND_PRICE)
```

See also

- "IAdxConvBond Interface" on page 190
- "AdxAttrConvBond" on page 305

AdxCorrelation

The `AdxCorrelation` component object implements the `IAdxCorrelation` interface that inherits its properties and methods from `IAdxObject`.

VBA sample

This example shows how to define a correlation matrix.

```
‘Control declaration and initialization
 Dim Correlation As AdxCorrelation
 Set Correlation = new AdxCorrelation
‘Variable declaration
 Dim CorrelationArray As Variant
‘Set correlation matrix
 Redim Preserve CorrelationArray (0 To 1, 0 To 1)
 CorrelationArray (0,0) = 0.19
 CorrelationArray (0,1) = -0.5
 CorrelationArray (1,0) = -0.5
 CorrelationArray (1,1) = 0.0725
 Correlation.SetAttribute AdxAttrCorrelation.ADX_ATTR1R_COR_MATRIX,
CorrelationArray
```

See also

- "AdxAttrCorrelation" on page 306
- "IAdxModelBuilder Interface: CreateVolatilityModel" on page 176

AdxCrossCurrency

The `AdxCrossCurrency` component object implements the `IAdxCrossCurrency` interface that inherits its properties and methods from `IAdxInstrument`. These instruments are traded on the Foreign Exchange Market and represent the cross currencies.

VBA sample

This example shows how to use the crosscurrency object assuming you have already defined the currency1, currency2 and currencyPivot of your cross.

```
'Control declaration and initialization
 Dim CrossCurrency As AdxCurrency
 Set CrossCurrency = new AdxCurrency
 Dim CalcMethod As AdxCalcMethod
 Set CalcMethod = new AdxCalcMethod
'Variable declaration
 Dim CrossCurrencyStyle As Variant
 Dim CalcDate As Variant
 Dim CalcStructure As Variant
'Definitions of CrossCurrency attributes to calculate
 Dim BidRes As Variant
 Dim AskRes As Variant
'Set CrossCurrency attributes and parameters
'Currency1 is GBP and Currency2 is EUR
 CrossCurrency.ErrorMode = DialogBox
 CrossCurrencyStyle = "GBPEUR"
 CalcDate = "22JUN99"
 CrossCurrency.Init CrossCurrencyStyle
 CrossCurrency.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_SETTLEMENT,
CalcDate  CrossCurrency.SetAttribute AdxAttrCrossCurrency.ADX_ATTR1E_CROSS_
QM1, Qm1
 CrossCurrency.SetAttribute AdxAttrCrossCurrency.ADX_ATTR1E_CROSS_QM2, Qm2
```

```
'Calculation initialization
'Set calculation formula
 CalcMethod.ErrorMode = DialogBox
 CalcStructure = "CMT:FORM"
 CalcMethod.Init (CalcStructure)
'Ask the formula pricer to compute cross values
'The AskToCompute method is called for each attribute to calculate
 CrossCurrency.AskToCompute AdxAttrCrossCurrency.ADX_ATTR3F_FX_CROSSA
 CalcMethod.AttachInstrument CrossCurrency
'Level #3 attributes computation
 CalcMethod.Compute
'Get the computed value
 BidRes = CrossCurrency.GetAttribute  AdxAttrCrossCurrency.ADX_ATTR3F_CROSS_
SPOT12_BID
 AskRes = CrossCurrency.GetAttribute  AdxAttrCrossCurrency.ADX_ATTR3F_CROSS_
SPOT12_ASK
```

See also

- "IAdxCrossCurrency Interface" on page 192
- "AdxAttrCrossCurrency" on page 306

AdxCurrency

The `AdxCurrency` component object implements the `IAdxCurrency` interface that inherits its properties and methods from `IAdxInstrument`. This object represents a currency with its spot rate.

VBA sample

This example shows how to define a currency with its Bid/Ask yield curve.

```
'Control declaration and initialization
 Dim Currency As AdxCurrency
 Set Currency = new AdxCurrency
 Dim RateModelBid As AdxRateModel
 Set RateModelBid = new AdxRateModel
 Dim RateModelAsk As AdxRateModel
 Set RateModelAsk = new AdxRateModel
'Variable declaration
 Dim CurrencyStyle As Variant
 Dim RateStructure As Variant
 Dim SpotBid As Variant
 Dim SpotAsk As Variant
'Set Currency attributes and parameters
 Currency.ErrorMode = DialogBox
 CurrencyStyle = "GBP"
 SpotBid = 1.416
 SpotAsk = 1.417
 Currency.Init CurrencyStyle
 Currency.SetAttribute AdxAttrCurrency.ADX_ATTR1F_CUR_SPOT_BID, SpotBid
 Currency.SetAttribute AdxAttrCurrency.ADX_ATTR1F_CUR_SPOT_ASK, SpotAsk

'Set RateModel attributes and parameters
 RateModelBid.ErrorMode = DialogBox
 RateStructure = "RM:YTM"
 RateModelBid.Init (RateStructure)
 RateModelBid.SetAttribute AdxAttrRateModel.ADX_ATTR1R_RATEMODEL_ARRAY, 0.05
 RateModelAsk.ErrorMode = DialogBox
 RateStructure = "RM:YTM"
 RateModelAsk.Init (RateStructure)
 RateModelAsk.SetAttribute AdxAttrRateModel.ADX_ATTR1R_RATEMODEL_ARRAY, 0.052
'Attach the RateModels to the Currency object
 Currency.AttachModel ADX_PTR_MODEL_DISCOUNTRATE_BID, RateModelBid
 Currency.AttachModel ADX_PTR_MODEL_DISCOUNTRATE_ASK, RateModelAsk
```

See also

- "IAdxCurrency Interface" on page 193
- "AdxAttrCurrency" on page 308

AdxDefault

The `AdxDefault` component object implements the `IAdxDefault` interface that inherits its properties and methods from `IAdxObject`.

See also

- "IAdxDefault Interface" on page 193

AdxDigitalCapFloor

The `AdxDigitalCapFloor` component object implements the `IAdxDigitalCapFloor` interface that inherits its properties and methods from `IAdxCapFloor` and `IAdxFloatLeg`. Digital caps and floors are interest rate risk management products based on All or Nothing options.

VBA sample

```
'Control declaration and initialization
 Dim DigitalCapFloor As AdxDigitalCapFloor
 Set DigitalCapFloor = new AdxDigitalCapFloor
 Dim RateModel As AdxRateModel
 Set RateModel = new AdxRateModel
 Dim CalcMethod As AdxCalcMethod
 Set CalcMethod = new AdxCalcMethod
'Variable declaration
 Dim CapFloorStructure As String
 Dim RateStructure As String
 Dim CalcStructure As String
 Dim CalcDate As Variant
 Dim StartDate As Variant
 Dim MaturityDate As Variant
 Dim CapStrike As Double
 Dim FloorStrike As Double
 Dim FirstRate As Double
 Dim RebateCap As Variant
 Dim RebateFloor As Variant
'Definitions of DigitalCapFloor attributes to calculate
 Dim Premium As Variant
'Set DigitalCapFloor attributes and parameters
 DigitalCapFloor.ErrorMode = DialogBox
 CapFloorStructure = "CAP FRQ:4 REFDATE:MAT CLDR:FRA1 CCM:MMA0 CFADJ:YES
DMC:F"
 CalcDate = "01OCT02"
```

```
 MaturityDate = "25APR05"
 FirstRate = 0.0
 CapStrike = 0.032
 FloorStrike = 0.032
 RebateCap = 0.02
 RebateFloor = 0.02
 DigitalCapFloor.Init(CapFloorStructure)
 DigitalCapFloor.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_SETTLEMENT,
CalcDate
 DigitalCapFloor.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_ISSUE,
IssueDate
 DigitalCapFloor.SetAttribute AdxAttrLeg.ADX_ATTR1S_LEG_MATURITY_CODE,
MaturityDate
 DigitalCapFloor.SetAttribute AdxAttrFloatLeg.ADX_ATTR1F_FLOATLEG_CAP,
CapStrike
 DigitalCapFloor.SetAttribute AdxAttrFloatLeg.ADX_ATTR1F_FLOATLEG_FLOOR,
FloorStrike
 DigitalCapFloor.SetAttribute AdxAttrFloatLeg.ADX_ATTR1F_FLOATLEG_CURRENT_
INDEX,  FirstRate
 DigitalCapFloor.SetAttribute  AdxAttrDigitalCapFloor.ADX_ATTR1R_DIGITAL_CAP_
REBATE, RebateCap
 DigitalCapFloor.SetAttribute  AdxAttrDigitalCapFloor.ADX_ATTR1R_DIGITAL_
FLOOR_REBATE, RebateFloor
'Set RateModel attributes and parameters
 RateModel.ErrorMode = DialogBox
 RateStructure = "RM:BS ZCTYPE:RATE DCB:A5 RATEFRQ:ZERO IM:LIN RATETYPE:ACT
CLDRADJ:CLDR"
 StartDate = "25OCT02"
 Dim RateArray as Variant
 ReDim Preserve RateArray (0 To 9, 0 To 1)
 RateArray (0, 0) = "01OCT02"
 RateArray (1, 0) = "02OCT02"
 RateArray (2, 0) = "03OCT02"
 RateArray (3, 0) = "04OCT03"
```

```
 RateArray (4, 0) = "12DEC03"
 RateArray (5, 0) = "03OCT05"
 RateArray (6, 0) = "03DEC05"
 RateArray (7, 0) = "15DEC05"
 RateArray (8, 0) = "17JAN07"
 RateArray (9, 0) = "15MAY07"
 RateArray (0, 1) = 0.0337
 RateArray (1, 1) = 0.03375
 RateArray (2, 1) = 0.032
 RateArray (3, 1) = 0.033756
 RateArray (4, 1) = 0.0331
 RateArray (5, 1) = 0.02
 RateArray (6, 1) = 0.02
 RateArray (7, 1) = 0.02
 RateArray (8, 1) = 0.04
 RateArray (9, 1) = 0.041
 RateModel.Init (RateStructure)
 RateModel.SetAttribute AdxAttrRateModel.ADX_ATTR1R_RATEMODEL_ARRAY,
RateArray
'Attach the RateModel to the CapFloor object
 DigitalCapFloor.AttachModel ADX_PTR_MODEL_DISCOUNTRATE, RateModel
'Calculation initialization
'Set calculation formula
 CalcMethod.ErrorMode = DialogBox
 CalcStructure = "CMT:FORM FT:BS"
 CalcMethod.Init (CalcStructure)
'Ask the formula pricer to compute the calculation attributes of your capfloor
'Instrument passed to the calculation method
'The AskToCompute method is called for each attribute to calculate
 DigitalCapFloor.AskToCompute(AdxAttrCapFloor.ADX_ATTR3F_CAPFLOOR_PREMIUM)
 CalcMethod.AttachInstrument DigitalCapFloor
'Level #3 attributes computation
 CalcMethod.Compute
'Get the computed value
 Premium =  DigitalCapFloor.GetAttribute(AdxAttrCapFloor.ADX_ATTR3F_CAPFLOOR_
PREMIUM)
```

See also

- "IAdxDigitalCapFloor Interface" on page 194
- "AdxAttrDigitalCapFloor" on page 309

AdxDividendModel

The `AdxDividendModel` component object implements the `IAdxDividendModel` interface that inherits its properties and methods from `IAdxModel`. This object describes the model used to provide the expected value of asset dividends.

VBA sample

See "AdxVolatilityModel" on page 294 to see how to use `AdxDividendModel`.

See also

- "IAdxDividendModel Interface" on page 195
- "AdxAttrDividendModel" on page 309

AdxFixedLeg

The `AdxFixedleg` component object implements the `IAdxFixedLeg` interface that inherits its properties and methods from `IAdxLeg`. This object describes fixed-leg-based instruments, such as bonds, convertibles, and index-linked bonds.

VBA sample

See "AdxBond" on page 235 to see how to use `AdxFixedLeg`.

See also

- "IAdxFixedLeg Interface" on page 196
- "AdxAttrFixedLeg" on page 310
- "AdxDateMovingConvention" on page 357
- "AdxEndOfMonthConvention" on page 363

AdxFloatLeg

The `AdxFloatLeg` component object implements the `IAdxFloatLeg` interface that inherits its properties and methods from `IAdxFloatLeg`. This object describes a float-leg-based instrument, such as an FRN.

VBA sample

See "AdxFrn" on page 260 to see how to use `AdxFloatLeg`.

See also

- "IAdxFloatLeg Interface" on page 197
- "AdxAttrFloatLeg" on page 311
- "AdxDateMovingConvention" on page 357
- "AdxEndOfMonthConvention" on page 363

AdxForex

The `AdxForex` component object implements the `IAdxForex` interface that inherits its properties and methods from `IAdxInstrument`.

VBA sample

This example shows how to create a forex object.

```
  Dim Forex As AdxForex
  Set Forex = new AdxForex
 'Variable declaration
  Dim InstrumentCode As Variant
 'Set Forex object
  Forex.ErrorMode = DialogBox
  InstrumentCode = "GBP7M="
  Forex.Init InstrumentCode
```

See also

- "IAdxForex Interface" on page 197
- "AdxAttrForex" on page 312

AdxFra

The `AdxFra` component object implements the `IAdxFra` interface that inherits its properties and methods from `IAdxInstrument`.

A forward rate agreements or FRA is a contract between two parties that fixes an interest rate for an agreed future lending period.

VBA sample

```
'This example shows how to compute a forward rate from a zero coupon curve:
'Control declaration and initialization
 Dim FRA As AdxFra
 Set FRA = new AdxFra
 Dim RateModel As AdxRateModel
 Set RateModel = new AdxRateModel
 Dim CalcMethod As AdxCalcMethod
 Set CalcMethod = new AdxCalcMethod
'Variable declaration
 Dim FRAStructure As Variant
 Dim RateStructure As Variant
 Dim CalcStructure As Variant
 Dim FRAStartDate As Variant
 Dim FRAPeriod As Variant
 Dim ZcArray As Variant
'Definitions of FRA attributes to calculate
 Dim ForwardRate As Variant
'Set FRA attributes and parameters
 FRA.ErrorMode = DialogBox
 FRAStructure = "CLDR:UKG CUR:GBP"
 FRAStartDate = "10JAN01"
 FRAPeriod = "3M"
 FRA.Init(FRAStructure)
 FRA.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_SETTLEMENT, FRAStartDate
 FRA.SetAttribute AdxAttrFRA.ADX_ATTR1S_FRA_PERIOD, FRAPeriod
```

```
'Set RateModel attributes and parameters
 RateModel.ErrorMode = DialogBox
 RateStructure = "RM:YC RATETYPE:CMP DCB:AA RATEFRQ:1"
 YcStartDate = "10OCT00"
 ReDim Preserve ZcArray (0 To 2, 0 To 5)
 ZcArray (0, 0) = "22JUN99"
 ZcArray (1, 0) = "22JUN00"
 ZcArray (2, 0) = "22JUN01"
 ZcArray (3, 0) = "22JUN02"
 ZcArray (4, 0) = "22JUN03"
 ZcArray (0, 1) = 0.03
 ZcArray (1, 1) = 0.035
 ZcArray (2, 1) = 0.04
 ZcArray (3, 1) = 0.045
 ZcArray (4, 1) = 0.05
 RateModel.Init (RateStructure)
 RateModel.SetAttribute AdxAttrRateModel.ADX_ATTR1R_RATEMODEL_ARRAY, ZcArray
'Attach the RateModel to the FRA object
 FRA.AttachModel ADX_PTR_MODEL_DISCOUNTRATE, RateModel
'Calculation initialization:
'Set calculation formula
 CalcMethod.ErrorMode = DialogBox
 CalcStructure = "CMT:FORM"
 CalcMethod.Init (CalcStructure)
'Ask the formula pricer to compute the forward rate
'The AskToCompute method is called for each attribute to calculate
 FRA.AskToCompute(AdxAttrFRA.ADX_ATTR3F_FRA_FORWARDRATE)
 CalcMethod.AttachInstrument FRA
'Level #3 attributes computation
 CalcMethod.Compute
'Get the computed value
 ForwardRate = FRA.GetAttribute(AdxAttrFRA.ADX_ ATTR3F_FRA_FORWARDRATE)
```

See also

- "IAdxFra Interface" on page 198
- "AdxAttrFra" on page 316

AdxFrn

The `AdxFrn` component object implements the `IAdxFrn` interface that inherits its properties and methods from `IAdxFloatLeg`.

FRNs or Floating Rate Notes are bonds that pay a variable interest rate.

```
'Control declaration and initialization
 Dim Frn As AdxFrn
 Set Frn= new AdxFrn
 Dim DiscountRateModel As AdxRateModel
 Set DiscountRateModel = new AdxRateModel
 Dim IndexRateModel As AdxRateModel
 Set IndexRateModel = new AdxRateModel
 Dim CalcMethod As AdxCalcMethod
 Set CalcMethod = new AdxCalcMethod
'Variable declaration
 Dim FrnStructure As Variant
 Dim DiscountRateStructure As Variant
 Dim IndexRateStructure As Variant
 Dim CalcStructure As Variant
 Dim CalcDate As Variant
 Dim MaturityDate As Variant
 Dim IssueDate As Variant
 Dim QuotedMargin As Variant
 Dim Libor As Variant
'Definitions of frn attributes to calculate
 Dim Price As Variant
```

```
'Set Frn attributes and parameters
  Frn.ErrorMode = DialogBox
 FrnStructure = "FRQ:2 CCM:AA"
 CalcDate = "10JAN01"
 MaturityDate = "25JUL10"
 IssueDate = "25MAR00"
 QuotedMargin = 0.0625
 Libor = 0.03
 Frn.Init(FrnStructure)
 Frn.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_SETTLEMENT, CalcDate
 Frn.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_ISSUE, IssueDate
 Frn.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_MATURITY, MaturityDate
 Frn.SetAttribute AdxAttrFloatLeg.ADX_ATTR1R_FLOATLEG_INDEX_SCENARIO, Libor
 Frn.SetAttribute AdxAttrFloatLeg.ADX_ATTR1F_FLOATLEG_SPREAD, QuotedMargin
'Set RateModel attributes and parameters
 DiscountRateModel.ErrorMode = DialogBox
 DiscountRateStructure = "RM:YTM RATETYPE:CMP DCB:AA RATEFRQ:1"
 YcStartDate = "10OCT00"
 Yield = 0.0475
 DiscountRateModel.Init (RateStructure)
 DisountRateModel.SetAttribute AdxAttrRateModel.ADX_ATTR1R_RATEMODEL_ARRAY,
Yield
 IndexRateModel.ErrorMode = DialogBox
 IndexRateStructure = "RM:YTM RATETYPE:CMP DCB:AA RATEFRQ:1"
 YcStartDate = "10OCT00"
 IndexYield = 0.049
 IndexRateModel.Init(RateStructure)
 IndexRateModel.SetAttribute AdxAttrRateModel.ADX_ATTR1R_RATEMODEL_ARRAY,
IndexYield
```

```
'Attach the RateModels to the Frn object
 Frn.AttachModel ADX_PTR_MODEL_DISCOUNTRATE, DiscountRateModel
 Frn.AttachModel ADX_PTR_MODEL_INDEXRATE, IndexRateModel
'Calculation initialization:
'Set calculation formula
 CalcMethod.ErrorMode = DialogBox
 CalcStructure = "CMT:FORM"
 CalcMethod.Init(CalcStructure)
'Ask the formula pricer to compute the calculation attributes of your bond
'Instrument passed to the calculation method
'The AskToCompute method is called for each attribute to calculate
 Frn.AskToCompute(AdxAttrFrn.ADX_ATTR3F_FRN_GROSS_PRICE)
 CalcMethod.AttachInstrument Frn
'Level #3 attributes computation
 CalcMethod.Compute
'Get the computed value
 Price = Frn.GetAttribute(AdxAttrFrn.ADX_ATTR3F_FRN_GROSS_PRICE)
```

See also

- "IAdxFrn Interface" on page 199
- "AdxAttrFrn" on page 317
- "AdxDateMovingConvention" on page 357
- "AdxEndOfMonthConvention" on page 363

AdxFrq

The `AdxFrq` component object implements the `IAdxFrq` interface that inherits its properties and methods from `IAdxObject`. `AdxFrq` provides the frequency description.

See also

- "IAdxObject Interface" on page 167
- "AdfinX Analytics Object Overview" on page 228
- "AdxFrequency" on page 366
- "AdxFrequencyType" on page 367

AdxFuture

The `AdxFuture` component object implements the `IAdxFuture` interface that inherits its properties and methods from `IAdxInstrument`.

A financial futures contract fixes the price and conditions for the delivery of a pre-defined financial instrument on a specific future date, at the time of the deal.

VBA sample

```
'Control declaration and initialization
 Dim Future As AdxFuture
 Set Future = new AdxFuture
 Dim Bond As AdxBond
 Set Bond = new AdxBond
 Dim RateModel As AdxRateModel
 Set RateModel = new AdxRateModel
 Dim CalcMethod As AdxCalcMethod
 Set CalcMethod = new AdxCalcMethod
'Variable declaration
 Dim FutureStructure As Variant
 Dim BondStructure As Variant
 Dim RateStructure As Variant
 Dim CalcStructure As Variant
 Dim CalcDate As Variant
 Dim FutMaturityCode As Variant
 Dim BondMaturityDate As Variant
 Dim Coupon As Variant
 Dim IssueDate As Variant
 Dim Yield As Variant
'Definitions of future attributes to calculate
 Dim ConvFactor As Variant
```

```
'Set Future attributes and parameters
 Future.ErrorMode = DialogBox
 FutureStructure = "RATE:6% RRTYPE:MMBA0 CUR:EUR BOND:BF_EUREX CRD:10C"
 CalcDate = "10JAN01"
 FutMaturityCode = "H01"
 Future.Init(FutureStructure)
 Future.SetAttribute AdxAttrFuture.ADX_ATTR1S_FUTURE_MATURITYCODE,
FutMaturityCode
'Set Bond attributes and parameters
 Bond.ErrorMode = DialogBox
 BondStructure = "FRQ:1 ACC:AA"
 CalcDate = "10JAN01"
 MaturityDate = "25JUL10"
 IssueDate = "25MAR00"
 Coupon = 0.05
 Bond.Init (BondStructure)
 Bond.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_SETTLEMENT, CalcDate
 Bond.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_ISSUE, IssueDate
 Bond.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_MATURITY, MaturityDate
 Bond.SetAttribute AdxAttrFixedLeg.ADX_ATTR2F_FIXEDLEG_COUPON, Coupon
'Set RateModel attributes and parameters
 RateModel.ErrorMode = DialogBox
 RateStructure = "RM:YTM RATETYPE:CMP DCB:AA RATEFRQ:1"
 YcStartDate = "10OCT00"
 Yield = 0.0475
 RateModel.Init (RateStructure)
 RateModel.SetAttribute AdxAttrRateModel.ADX_ATTR1R_RATEMODEL_ARRAY, Yield
'Attach the RateModel to the Bond object
 Bond.AttachModel ADX_PTR_MODEL_DISCOUNTRATE, RateModel
'Attach the Bond to the Future object
 Future.AttachModel ADX_PTR_MODEL_DISCOUNTRATE, RateModel
 Future.AttachInstrument ADX_PTR_INSTRUMENT_UNDERLYING1, Bond
```

```
'Calculation initialization:
'Set calculation formula
 CalcMethod.ErrorMode = DialogBox
 CalcStructure = "CMT:FORM"
 CalcMethod.Init (CalcStructure)
'Ask the formula pricer to compute the calculation attributes of your bond
'Instrument passed to the calculation method
'The AskToCompute method is called for each attribute to calculate
 Future.AskToCompute(AdxAttrFuture.ADX_ATTR3F_FUTURE_CONVFACTOR)
 CalcMethod.AttachInstrument Future
'Level #3 attributes computation
 CalcMethod.Compute
'Get the computed value
 ConvFactor = Future.GetAttribute(AdxAttrFuture.ADX_ATTR3F_FUTURE_CONVFACTOR)
```

See also

- "IAdxFuture Interface" on page 201
- "AdxAttrFuture" on page 318

AdxFxModel

The `AdxFxModel` component object implements the `IAdxFxModel` interface that inherits its properties and methods from `IAdxModel`.

VBA sample

See AdxForex to see how to use `AdxFxModel`.

See also

- "IAdxFxModel Interface" on page 202
- "AdxAttrFxModel" on page 319

AdxIlb

The `AdxIlb` component object implements the `IAdxIlb` interface that inherits its properties and methods from `IAdxFixedLeg`. An index-linked bond is similar to a conventional fixed-rate bond, except that all cashflows (accrued interest, coupon, and principal at redemption) are multiplied by a coefficient based upon the change in the inflation reference index between the base date and the payment date.

VBA sample

```
'Control declaration and initialization
 Dim Ilb As AdxIlb
 Set Ilb = new AdxIlb
 Dim RateModel As AdxRateModel
 Set RateModel = new AdxRateModel
 Dim CalcMethod As AdxCalcMethod
 Set CalcMethod = new AdxCalcMethod
'Variable declaration
 Dim IlbStructure As Variant
 Dim RateStructure As Variant
 Dim CalcStructure As Variant
 Dim CalcDate As Variant
 Dim MaturityDate As Variant
 Dim Coupon As Variant
 Dim IssueDate As Variant
 Dim Inflation As Variant
 Dim Yield As Variant
'Definitions of Ilb attributes to calculate
 Dim Price As Variant
```

```
'Set Ilb attributes and parameters
 Ilb.ErrorMode = DialogBox
 IlbStructure = "FRQ:2 ACC:AA"
 CalcDate = "10JAN01"
 MaturityDate = "25JUL10"
 IssueDate = "25MAR00"
 Coupon = 0.05
 Inflation = 0.03
 Ilb.Init (BondStructure)
 Ilb.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_SETTLEMENT, CalcDate
 Ilb.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_ISSUE, IssueDate
 Ilb.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_MATURITY, MaturityDate
 Ilb.SetAttribute AdxAttrIlb.ATTR1R_ILB_INFLATION_RATEARRAY, Inflation
 Ilb.SetAttribute AdxAttrFixedLeg.ADX_ATTR2F_FIXEDLEG_COUPON, Coupon
'Set RateModel attributes and parameters
 RateModel.ErrorMode = DialogBox
 RateStructure = "RM:YTM RATETYPE:CMP DCB:AA RATEFRQ:1"
 YcStartDate = "10OCT00"
 Yield = 0.0475
 RateModel.Init (RateStructure)
 RateModel.SetAttribute AdxAttrRateModel.ADX_ATTR1R_RATEMODEL_ARRAY, Yield
'Attach the RateModel to the Bond object
 Ilb.AttachModel ADX_PTR_MODEL_DISCOUNTRATE, RateModel
'Calculation initialization:
'Set calculation formula
 CalcMethod.ErrorMode = DialogBox
 CalcStructure = "CMT:FORM"
 CalcMethod.Init (CalcStructure)

'Ask the formula pricer to compute the calculation attributes of your bond
'Instrument passed to the calculation method
'The AskToCompute method is called for each attribute to calculate
 Ilb.AskToCompute (AdxAttrBond.ADX_ATTR3F_BOND_PRICE)
 CalcMethod.AttachInstrument Ilb
'Level #3 attributes computation
 CalcMethod.Compute
'Get the computed value
 Price = Ilb.GetAttribute(AdxAttrBond.ADX_ATTR3F_BOND_PRICE)
```

See also

- "IAdxIlb Interface" on page 204
- "AdxAttrIlb" on page 320

- "AdxDateMovingConvention" on page 357
- "AdxEndOfMonthConvention" on page 363

AdxInit

The `AdxInit` component object implements the `IAdxInit` interface that inherits its properties and methods from `IAdxInit`. It is used to overload the Adfin default parameter.

See also

- "AdfinX Analytics Interfaces" on page 167

AdxLibor

The `AdxLibor` component object implements the `IAdxLibor` interface which inherits its properties and methods from "IAdxObject Interface" on page 167.

See also

"IAdxLibor Interface" on page 205

AdxMapLibor

The `AdxMapLibor` component object implements the `IAdxMapLibor` interface which inherits its properties and methods from "IAdxObject Interface" on page 167.

See also

"IAdxMapLibor Interface" on page 206

AdxModelBuilder

The `AdxModelBuilder` component object implements the `IAdxModelBuilder` interface which inherits its properties and methods from "IAdxObject Interface" on page 167.

See also

"IAdxModelBuilder Interface" on page 173

AdxNToDefaultCDS

The `AdxNToDefaultCDS` component object implements the `IAdxNToDefaultCDS` interface that inherits its properties and methods from `IAdxOption`.

See also

- "AdfinX Analytics Interfaces" on page 167
- "AdfinX Analytics Parameters and Constants" on page 295

AdxOpBinary

The `AdxOpBinary` component object implements the `IAdxOpBinary` interface that inherits its properties and methods from `IAdxOption`.

One-touch binary options generally pay a pre-determined fixed amount of cash so long as the option was in the money at some stage during its life. The payoff can either be immediate when the underlying asset is in the money or deferred until the expiry date. Similarly, a no-touch binary option pays off a fixed amount of cash if the option remains out-of-the-money during its lifetime.

VBA sample

```
'Control declaration and initialization
  Dim OpBinary As AdxOpBinary
  Set OpBinary = new AdxOpBinary
  Dim Asset As AdxAsset
  Set Asset = new AdxAsset
  Dim RateModel As AdxRateModel
  Set RateModel = new AdxRateModel
  Dim VolatilityModel As AdxVolatilityModel
  Set VolatilityModel = new AdxVolatilityModel
  Dim CalcMethod As AdxCalcMethod
  Set CalcMethod = new AdxCalcMethod
'Variable declaration
  Dim OptionStructure As Variant
  Dim UnderlyingStructure As Variant
  Dim RateStructure As Variant
  Dim CalcStructure As Variant
  Dim CalcDate As Variant
  Dim SpotPrice As Variant
  Dim Volatility As Variant
  Dim RiskFreeRate As Variant
  Dim ExpiryDate As Variant
  Dim StrikePrice As Variant
```

```
'Definitions of option attributes to calculate
 Dim Premium As Variant
'Set Asset attributes and parameters
 SpotPrice = 134
 UnderlyingStructure = "UI:SEC"
 Asset.Init(UnderlyingStructure)
 Asset.SetAttribute AdxAttrAsset.ADX_ATTR1F_ASSET_PRICE, SpotPrice
'Set VolatilityModel attributes
 VolatilityModel.ErrorMode = DialogBox
 Volatility = 0.18
 VolatilityModel.SetAttribute AdxAttrVolatilityModel.ADX_ATTR1R_VOLATILITY,
Volatility
'Attach the Volatility to the Asset object
 Asset.AttachModel ADX_PTR_MODEL_DIVIDEND, VolatilityModel
'Set option attributes and parameters
 OpBinary.ErrorMode = DialogBox
 OptionStructure = "CALL EXM:A"
 CalcDate = "10JAN01"
 ExpiryDate = "25JUL02"
 StrikePrice = 136
 OpBinary.Init (OptionStructure)
 OpBinary.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_SETTLEMENT, CalcDate
 OpBinary.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_MATURITY, ExpiryDate
 OpBinary.SetAttribute AdxAttrOpBinary.ADX_ATTR1F_EXO_CASH_AMOUNT, Cash
Amount
 OpBinary.SetAttribute AdxAttrOption.ADX_ATTR1F_OPT_STRIKE, StrikePrice
'Set RateModel attributes and parameters
 RateModel.ErrorMode = DialogBox
 RateStructure = "RM:YTM RATETYPE:CONT"
 RiskFreeRate = 0.0475
 RateModel.Init (RateStructure)
 RateModel.SetAttribute AdxAttrRateModel.ADX_ATTR1R_RATEMODEL_ARRAY,
RiskFreeRate
```

```
'Attach the RateModel to the Option object
 OpBinary.AttachModel ADX_PTR_MODEL_DISCOUNTRATE, RateModel
 OpBinary.AttachInstrument ADX_PTR_INSTRUMENT_UNDERLYING1, Asset
'Calculation initialization:
'Set calculation formula
 CalcMethod.ErrorMode = DialogBox
 CalcStructure = "CMT:FORM"
 CalcMethod.Init (CalcStructure)
'Ask the formula pricer to compute the calculation attributes of your bond
'Instrument passed to the calculation method
'The AskToCompute method is called for each attribute to calculate
 OpBinary.AskToCompute (AdxAttrOption.ADX_ATTR3F_OPT_PREMIUM)
 CalcMethod.AttachInstrument OpBinary
'Level #3 attributes computation
 CalcMethod.Compute
'Get the computed value
 Premium = OpBinary.GetAttribute(AdxAttrOption.ADX_ATTR3F_OPT_PREMIUM)
```

See also

- "IAdxOpBinary Interface" on page 208
- "AdxAttrOpBinary" on page 325

AdxOpLookBack

The `AdxOpLookBack` component object implements the `IAdxOpLookBack` interface that inherits its properties and methods from `IAdxOption`.

Lookback options are based on the optimum price observed for the underlying instrument. They allow the holder to choose at expiry for strike price the best price achieved during the life of the option.

VBA sample

```
'Control declaration and initialization
 Dim OpLookBack As AdxOpLookBack
 Set OpLookBack = new AdxOpLookBack
 Dim Asset As AdxAsset
 Set Asset = new AdxAsset
 Dim RateModel As AdxRateModel
 Set RateModel = new AdxRateModel
 Dim VolatilityModel As AdxVolatilityModel
 Set VolatilityModel = new AdxVolatilityModel
 Dim CalcMethod As AdxCalcMethod
 Set CalcMethod = new AdxCalcMethod
'Variable declaration
 Dim OptionStructure As Variant
 Dim UnderlyingStructure As Variant
 Dim RateStructure As Variant
 Dim CalcStructure As Variant
 Dim CalcDate As Variant
 Dim SpotPrice As Variant
 Dim Volatility As Variant
 Dim RiskFreeRate As Variant
 Dim ExpiryDate As Variant
 Dim StrikePrice As Variant
'Definitions of option attributes to calculate
 Dim Premium As Variant
```

```
'Set Asset attributes and parameters
 SpotPrice = 134
 UnderlyingStructure = "UI:SEC"
 Asset.Init(UnderlyingStructure)
 Asset.SetAttribute AdxAttrAsset.ADX_ATTR1F_ASSET_PRICE, SpotPrice
'Set VolatilityModel attributes
 VolatilityModel.ErrorMode = DialogBox
 Volatility = 0.18
 VolatilityModel.SetAttribute AdxAttrVolatilityModel.ADX_ATTR1R_VOLATILITY,
Volatility
'Attach the Volatility to the Asset object
 Asset.AttachModel ADX_PTR_MODEL_DIVIDEND, VolatilityModel
 Set option attributes and parameters
 OpLookBack.ErrorMode = DialogBox
 OptionStructure = "CALL EXM:A"
 CalcDate = "10JAN01"
 ExpiryDate = "25JUL02"
 StrikePrice = 136
 OpLookBack.Init (OptionStructure)
 OpLookBack.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_SETTLEMENT,
CalcDate
 OpLookBack.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_MATURITY,
ExpiryDate
 OpLookBack.SetAttribute AdxAttrOpLookBack.ADX_ATTR1F_MINMAX, Price
 OpLookBack.SetAttribute AdxAttrOption.ADX_ATTR1F_OPT_STRIKE, StrikePrice
'Set RateModel attributes and parameters
 RateModel.ErrorMode = DialogBox
 RateStructure = "RM:YTM RATETYPE:CONT"
 RiskFreeRate = 0.0475
 RateModel.Init (RateStructure)
 RateModel.SetAttribute AdxAttrRateModel.ADX_ATTR1R_RATEMODEL_ARRAY,
RiskFreeRate
```

```
'Attach the RateModel to the Option object
 OpLookBack.AttachModel ADX_PTR_MODEL_DISCOUNTRATE, RateModel
 OpLookBack.AttachInstrument ADX_PTR_INSTRUMENT_UNDERLYING1, Asset'
'Calculation initialization:
'Set calculation formula
 CalcMethod.ErrorMode = DialogBox
 CalcStructure = "CMT:FORM"
 CalcMethod.Init (CalcStructure)
'Ask the formula pricer to compute the calculation attributes of your bond
'Instrument passed to the calculation method
'The AskToCompute method is called for each attribute to calculate:
 OpLookBack.AskToCompute (AdxAttrOption.ADX_ATTR3F_OPT_PREMIUM)
 CalcMethod.AttachInstrument OpBinary
'Level #3 attributes computation
 CalcMethod.Compute
'Get the computed value
 Premium = OpLookBack.GetAttribute(AdxAttrOption.ADX_ATTR3F_OPT_PREMIUM)
```

See also

- "IAdxOpLookBack Interface" on page 208
- "AdxAttrOpLookBack" on page 325

AdxOption

The `AdxOption` component object implements the `IAdxOption` interface that inherits its properties and methods from `IAdxInstrument`.

VBA sample

```
'Control declaration and initialization
 Dim Option As AdxOption
 Set Option = new AdxOption
 Dim Asset As AdxAsset
 Set Asset = new AdxAsset
 Dim RateModel As AdxRateModel
 Set RateModel = new AdxRateModel
 Dim VolatilityModel As AdxVolatilityModel
 Set VolatilityModel = new AdxVolatilityModel
 Dim CalcMethod As AdxCalcMethod
 Set CalcMethod = new AdxCalcMethod
'Variable declaration
 Dim OptionStructure As Variant
 Dim UnderlyingStructure As Variant
 Dim RateStructure As Variant
 Dim CalcStructure As Variant
 Dim CalcDate As Variant
 Dim SpotPrice As Variant
 Dim Volatility As Variant
 Dim RiskFreeRate As Variant
 Dim ExpiryDate As Variant
 Dim StrikePrice As Variant
'Definitions of option attributes to calculate
 Dim Premium As Variant
```

```
'Set Asset attributes and parameters
 SpotPrice = 134
 UnderlyingStructure = "UI:SEC"
 Asset.Init(UnderlyingStructure)
 Asset.SetAttribute AdxAttrAsset.ADX_ATTR1F_ASSET_PRICE, SpotPrice'
'Set VolatilityModel attributes
 VolatilityModel.ErrorMode = DialogBox
 Volatility = 0.18
 VolatilityModel.SetAttribute  AdxAttrVolatilityModel.ADX_ATTR1R_
VOLATILITY,Volatility
'Attach the Volatility to the Asset object
 Asset.AttachModel ADX_PTR_MODEL_DIVIDEND, VolatilityModel
'Set option attributes and parameters
 Option.ErrorMode = DialogBox
 OptionStructure = "CALL EXM:A"
 CalcDate = "10JAN01"
 ExpiryDate = "25JUL02"
 StrikePrice = 136
 Option.Init (OptionStructure)
 Option.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_SETTLEMENT, CalcDate
 Option.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_MATURITY, ExpiryDate
 Option.SetAttribute AdxAttrOption.ADX_ATTR1F_OPT_STRIKE, StrikePrice
'Set RateModel attributes and parameters
 RateModel.ErrorMode = DialogBox
 RateStructure = "RM:YTM RATETYPE:CONT"
 RiskFreeRate = 0.0475
 RateModel.Init (RateStructure)
 RateModel.SetAttribute AdxAttrRateModel.ADX_ATTR1R_RATEMODEL_ARRAY,
RiskFreeRate
'Attach the RateModel to the Option object
 Option.AttachModel ADX_PTR_MODEL_DISCOUNTRATE, RateModel
 Option.AttachInstrument ADX_PTR_INSTRUMENT_UNDERLYING1, Asset
```

```
'Calculation initialization:
'Set calculation formula
 CalcMethod.ErrorMode = DialogBox
CalcStructure = "CMT:FORM"
 CalcMethod.Init (CalcStructure)
'Ask the formula pricer to compute the calculation attributes of your bond
'Instrument passed to the calculation method
'The AskToCompute method is called for each attribute to calculate
 Option.AskToCompute (AdxAttrOption.ADX_ATTR3F_OPT_PREMIUM)
 CalcMethod.AttachInstrument Option
'Level #3 attributes computation
 CalcMethod.Compute
'Get the computed value
 Premium = Option.GetAttribute(AdxAttrOption.ADX_ATTR3F_OPT_PREMIUM)
```

See also

- "IAdxOption Interface" on page 209
- "AdxAttrOption" on page 325

AdxParse

The `AdxParse` component object implements the `IAdxParse` interface that inherits its properties and methods from `IAdxObject`.

This object allows you to parse a data string formatted in fraction or bid/ask format.

VBA sample

```
'This example shows how to use the parser object to parse string of market data:'Control declaration and initialization
 Dim Parser As AdxParse
 Set Parser = new AdxParse
'Variable declaration
 Dim ParserMode As Variant
 Dim MarketString As Variant
'Variable declaration for result array
 Dim ResultArray As Variant
'Set Parser attributes and parameters
 Parser.ErrorMode = DialogBox
 ParserMode = "PDF:BA LEN:10 POS:41 RET:1"
 MarketString = "ZDWN 19703 PORTUGA 7.700 07/06/05 AA- 116.28-47 4.975 11.1
<CDCEUROFRF8>"
 Parser.Init ParserMode
Parser.SetAttribute AdxAttrParse.ADX_ATTR1S_PARSE_STR, MarketString
'Get the result in an array
 ResultArray = Parser.GetAttribute AdxAttrParse.ADX_ATTR3R_PARSE_RESULT
```

See also

- "IAdxParse Interface" on page 210
- "AdxAttrParse" on page 327

AdxRainbow

The `AdxRainbow` component object implements the `IAdxRainbow` interface that inherits its properties and methods from `IAdxOption`.

Rainbow options are options where the final payoff is determined by the highest performance achieved at the expiration date by two or more underlying assets. Rainbow options can be either American or European options.

VBA sample

```
'Control declaration and initialization
 Dim Rainbow As AdxRainbow
 Set Rainbow = new AdxRainbow
 Dim Asset As AdxAsset
 Set Asset = new AdxAsset
 Dim RateModel As AdxRateModel
 Set RateModel = new AdxRateModel
 Dim VolatilityModel As AdxVolatilityModel
 Set VolatilityModel = new AdxVolatilityModel
 Dim CalcMethod As AdxCalcMethod
 Set CalcMethod = new AdxCalcMethod
'Variable declaration
 Dim OptionStructure As Variant
 Dim UnderlyingStructure As Variant
 Dim RateStructure As Variant
 Dim CalcStructure As Variant
 Dim CalcDate As Variant
 Dim SpotPrice As Variant
 Dim Volatility As Variant
 Dim RiskFreeRate As Variant
 Dim ExpiryDate As Variant
 Dim StrikePrice As Variant
'Definitions of option attributes to calculate
 Dim Premium As Variant
```

```
'Set Asset attributes and parameters
 SpotPrice = 134
 UnderlyingStructure = "UI:SEC"
 Asset.Init(UnderlyingStructure)
 Asset.SetAttribute AdxAttrAsset.ADX_ATTR1F_ASSET_PRICE, SpotPrice
'Set VolatilityModel attributes
 VolatilityModel.ErrorMode = DialogBox
 Volatility = 0.18
 VolatilityModel.SetAttribute AdxAttrVolatilityModel.ADX_ATTR1R_VOLATILITY,
Volatility
'Attach the Volatility to the Asset object
 Asset.AttachModel ADX_PTR_MODEL_DIVIDEND, VolatilityModel
'Set Option attributes and parameters
 Rainbow.ErrorMode = DialogBox
 OptionStructure = "CALL EXM:A"
 CalcDate = "10JAN01"
 ExpiryDate = "25JUL02"
 StrikePrice = 136
 Rainbow.Init (OptionStructure)
 Rainbow.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_SETTLEMENT, CalcDate
 Rainbow.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_MATURITY, ExpiryDate
 Rainbow.SetAttribute AdxAttrRainbow.ADX_ATTR1F_EXO_DOUBLESTRIKE, StrikeArray
 Rainbow.SetAttribute AdxAttrOption.ADX_ATTR1F_OPT_STRIKE, StrikePrice
'Set RateModel attributes and parameters
 RateModel.ErrorMode = DialogBox
 RateStructure = "RM:YTM RATETYPE:CONT"
 RiskFreeRate = 0.0475
 RateModel.Init (RateStructure)
 RateModel.SetAttribute AdxAttrRateModel.ADX_ATTR1R_RATEMODEL_ARRAY,
RiskFreeRate
```

```
'Attach the RateModel to the Option object
 Rainbow.AttachModel ADX_PTR_MODEL_DISCOUNTRATE, RateModel
 Rainbow.AttachInstrument ADX_PTR_INSTRUMENT_UNDERLYING1, Asset
'Calculation initialization:
'Set calculation formula
 CalcMethod.ErrorMode = DialogBox
 CalcStructure = "CMT:FORM"
 CalcMethod.Init (CalcStructure)
'Ask the formula pricer to compute the calculation attributes of your bond
'Instrument passed to the calculation method
'The AskToCompute method is called for each attribute to calculate:
 Rainbow.AskToCompute (AdxAttrOption.ADX_ATTR3F_OPT_PREMIUM)
 CalcMethod.AttachInstrument Rainbow
'Level #3 attributes computation
 CalcMethod.Compute
'Get the computed value
 Premium = Rainbow.GetAttribute(AdxAttrOption.ADX_ATTR3F_OPT_PREMIUM)
```

See also

- "IAdxRainbow Interface" on page 210
- "AdxAttrRainbow" on page 328

AdxRateModel

The `AdxRateModel` component object implements the `IAdxRateModel` interface that inherits its properties and methods from `IAdxModel`. This object describes the rate model used to value or price a financial product.

VBA sample

See AdxBond to see how to use `AdxRateModel`.

See also

- "IAdxRateModel Interface" on page 211
- "AdxAttrRateModel" on page 328

AdxRepo

The `AdxRepo` component object implements the `IAdxRepo` interface that inherits its properties and methods from `IAdxInstrument`. It represents a Repurchase agreement. The available collateral are bond and cashflows.

VBA sample

```
'Control declaration and initialization
  Dim Repo As AdxRepo
  Set Repo = new AdxRepo
  Dim Bond As AdxBond
  Set Bond = new AdxBond
  Dim RateModel As AdxRateModel
  Set RateModel = new AdxRateModel
  Dim RepoRateModel As AdxRateModel
  Set RepoRateModel = new AdxRateModel
  Dim CalcMethod As AdxCalcMethod
  Set CalcMethod = new AdxCalcMethod
'Variable declaration
  Dim RepoStructure As Variant
  Dim BondStructure As Variant
  Dim RateStructure As Variant
  Dim RepoRateStructure As Variant
  Dim CalcStructure As Variant
  Dim CalcDate As Variant
  Dim RepoMaturityDate As Variant
  Dim BondMaturityDate As Variant
  Dim Coupon As Variant
  Dim IssueDate As Variant
  Dim Npv As Variant
  Dim Fv As Variant
```

```
Dim ImpRate As Variant
 Dim Fv As Variant
 Dim Yield As Variant
‘Definitions of bond attributes to calculate
 Dim Npv As Variant‘
‘Set Repo attributes and parameters
 Repo.ErrorMode = DialogBox
 RepoStructure = "NPV:C "
 CalcDate = "10JAN01"
 RepoMaturityDate = "10FEB01"
 Fv = 1.002
 Npv = 0.99565
 Repo.Init(RepoStructure)
 Repo.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_SETTLEMENT, CalcDate
 Repo.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_MATURITY,
RepoMaturityDate
 Repo.SetAttribute AdxAttrrepo.ADX_ ATTR3F_REPO_FV, Fv
 Repo.SetAttribute AdxAttrrepo.ADX_ ATTR3F_REPO_NPV, Npv
‘Set Bond attributes and parameters
 Bond.ErrorMode = DialogBox
 BondStructure = "FRQ:1 ACC:AA"
 MaturityDate = "25JUL10"
 IssueDate = "25MAR00"
 Coupon = 0.05
 Bond.Init (BondStructure)
 Bond.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_SETTLEMENT, CalcDate
 Bond.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_ISSUE, IssueDate
 Bond.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_MATURITY, MaturityDate
 Bond.SetAttribute AdxAttrFixedLeg.ADX_ATTR2F_FIXEDLEG_COUPON, Coupon
‘Set RateModel attributes and parameters
 RateModel.ErrorMode = DialogBox
 RateStructure = "RM:YTM RATETYPE:CMP DCB:AA RATEFRQ:1"
```

```
YcStartDate = "10OCT00"
 Yield = 0.0475
RepoRateModel.SetAttribute AdxAttrRateModel.ADX_ATTR1R_RATEMODEL_ARRAY, Yield
RateModel.Init (RateStructure)
 RateModel.SetAttribute AdxAttrRateModel.ADX_ATTR1R_RATEMODEL_ARRAY, Yield
 RepoRateModel.ErrorMode = DialogBox
 RepoRateStructure = "RM:YTM RATETYPE:CMP DCB:AA RATEFRQ:1"
 YcStartDate = "10OCT00"
 Yield = 0.048
 RepoRateModel.Init (RepoRateStructure)
'Attach the RateModel to the Bond object
 Bond.AttachModel ADX_PTR_MODEL_DISCOUNTRATE, RateModel
'Attach the Bond to the Repo object
 Repo.AttachModel ADX_PTR_MODEL_REPO_RATE, RepoRateModel
 Repo.AttachInstrument ADX_ PTR_INSTRUMENT_COLLATERAL, Bond
'Calculation initialization:
'Set calculation formula
 CalcMethod.ErrorMode = DialogBox
 CalcStructure = "CMT:FORM"
 CalcMethod.Init (CalcStructure)
'Ask the formula pricer to compute the calculation attributes of your bond
'Instrument passed to the calculation method
'The AskToCompute method is called for each attribute to calculate
 Repo.AskToCompute(AdxAttrRepo.ATTR3F_REPO_REPO_RATE)
 CalcMethod.AttachInstrument Repo
'Level #3 attributes computation
 CalcMethod.Compute
'Get the computed value
 ImpRate = Repo.GetAttribute(AdxAttrRepo. ATTR3F_REPO_REPO_RATE)
```

See also

- "IAdxRepo Interface" on page 211
- "AdxAttrRepo" on page 330

AdxRiskModel

The `AdxRiskModel` component object implements the `IAdxRiskModel` interface that inherits its properties and methods from `IAdxModel`.

VBA sample

```
'Control declaration and initialization
 Dim RateModel As AdxRateModel
 Dim RateModel = new AdxRateModel
 Dim RiskModel As AdxRiskModel
 Set RiskModel = new AdxRiskModel
 Dim CalcMethod As AdxCalcMethod
 Set CalcMethod = new CalcMethod
'Variable declaration
 Dim Swap As AdxSwap
 Set Swap = new AdxSwap
 Dim SwapStructure As String
 Dim CalcStructure As String
 Dim RateStructure As String
 Dim RiskStructure As String
 Dim CalcDate As Variant
 Dim MaturityDate As Variant
 Dim Npv As Variant
'Definitions of swap attributes to calculate
 Dim Spread As Variant
'Set Swap attributes and parameters
 Swap.ErrorMode = DialogBox
 SwapStructure = "CDSTYPE:AMERCDS DMC:M CLDR:EMU_FI LFLOAT LFIXED FRQ:4
CCM:MMA0"
 CalcDate = "19SEP02"
MaturityDate = "19SEP07"
```

```
 Npv = 0
 Swap.Init(SwapStructure)
 Swap.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_SETTLEMENT, CalcDate
 Swap.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_MATURITY, MaturityDate
 Swap SetAttribute AdxAttrSwap.ADX_ATTR3F_SWAP_NPV, Npv
'Set RateModel attributes and parameters
 RateModel.ErrorMode = DialogBox
 RateStructure = "RM:YC ZCTYPE:DF IM:CUBD"
 Dim RateArray as Variant
 ReDim Preserve RateArray (0 To 9, 0 To 1)
 RateArray (0, 0) = "19SEP02"
 RateArray (1, 0) = "19SEP03"
 RateArray (2, 0) = "19SEP04"
 RateArray (3, 0) = "19SEP05"
 RateArray (4, 0) = "19SEP06"
 RateArray (5, 0) = "19SEP07"
 RateArray (6, 0) = "19SEP08"
 RateArray (7, 0) = "19SEP09"
 RateArray (8, 0) = "19SEP10"
 RateArray (9, 0) = "19SEP11"
 RateArray (0, 1) = 1
 RateArray (1, 1) = 0.96
 RateArray (2, 1) = 0.93
 RateArray (3, 1) = 0.89
 RateArray (4, 1) = 0.86
 RateArray (5, 1) = 0.82
 RateArray (6, 1) = 0.78
 RateArray (7, 1) = 0.74
 RateArray (8, 1) = 0.70
RateArray (9, 1) = 0.67
 RateModel.Init (RateStructure)
```

```
 RateModel.SetAttribute AdxAttrRateModel.ADX_ATTR1R_RATEMODEL_ARRAY,
RateArray
'Set RiskModel attributes and parameters
 RiskModel.ErrorMode = DialogBox
 RiskStructure = "RISKMODEL:CIR RECOVERY:0.3 NBDAYS:5 ND:DIS"
 Dim RiskArray as Variant
 ReDim Preserve RiskArray (0 To 2, 0 To 0)
 RiskArray (0, 0) = 0.04016
 RiskArray (1, 0) = 0.0101260797
 RiskArray (2, 0) = 0.10950
 RiskModel.Init (RiskStructure)
 RiskModel.SetAttribute AdxAttrRiskModel.ADX_ATTR1R_RISKMODEL_ARRAY,
RiskArray
'Attach the Models to the Swap
 Swap.AttachModel ADX_PTR_MODEL_DISCOUNTRATE, RateModel
 Swap.AttachModel ADX_PTR_MODEL_RISK, RiskModel
'Calculation initialization:
'Set calculation formula
 CalcMethod.ErrorMode = DialogBox
 CalcStructure = "CMT:FORM"
 CalcMethod.Init(CalcStructure)
'Ask the formula pricer to compute the calculation attributes of your bond
'Instrument passed to the calculation method
'The AskToCompute method is called for each attribute to calculate
 Swap.AskToCompute AdxAttrSwap.ADX_ATTR3F_SWAP_FIXED_RATE
 CalcMethod.AttachInstrument Swap
'Level #3 attributes computation
 CalcMethod.Compute
'Get the computed value
 Spread = Swap.GetAttribute(AdxAttrSwap.ADX_ATTR3F_SWAP_FIXED_RATE)
```

See also

- "IAdxRiskModel Interface" on page 213
- "AdxAttrRiskModel" on page 331

AdxSchedule

The `AdxSchedule` component object implements the `IAdxSchedule` interface that inherits its properties and methods from `IAdxInstrument`.

See also

- "AdfinX Analytics Interfaces" on page 167
- "AdfinX Analytics Parameters and Constants" on page 295

AdxSwap

The `AdxSwap` component object implements the `IAdxSwap` interface that inherits its properties and methods from `IAdxInstrument`. An `AdxSwap` object is built from its legs.

VBA sample

```
'Control declaration and initialization
  Dim Swap As AdxSwap
  Set Swap = new AdxSwap
  Dim FixedLeg As AdxFixedLeg
  Set FixedLeg = new AdxFixedLeg
  Dim FloatLeg As AdxFloatLeg
  Set FloatLeg = new AdxFloatLeg
  Dim DiscountRateModel As AdxRateModel
  Set DiscountRateModel = new AdxRateModel
  Dim IndexRateModel As AdxRateModel
  Set IndexRateModel = new AdxRateModel
  Dim CalcMethod As AdxCalcMethod
  Set CalcMethod = new AdxCalcMethod
'Variable declaration
  Dim SwapStructure As Variant
  Dim FixedLegStructure As Variant
  Dim FloatLegStructure As Variant
  Dim DiscountRateStructure As Variant
  Dim IndexRateStructure As Variant
  Dim CalcStructure As Variant
  Dim CalcDate As Variant
  Dim MaturityDate As Variant
  Dim IssueDate As Variant
```

```
 Dim FixedRate As Variant
 Dim Libor As Variant
'Definitions of frn attributes to calculate
 Dim Npv As Variant
'Set FixedLeg attributes and parameters
 FixedLeg.ErrorMode = DialogBox
 FixedStructure = "FRQ:1 ACC:AA"
 FixedRate = 0.0625
 FixedLeg.Init(FloatLegStructure)
 FixedLeg.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_SETTLEMENT, CalcDate
 FixedLeg.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_ISSUE, IssueDate
 FixedLeg.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_MATURITY,
MaturityDate
 FixedLeg.SetAttribute AdxAttrFixedLeg.ATTR2F_FIXEDLEG_COUPON, FixedRate
'Set FloatLeg attributes and parameters
 FloatLeg.ErrorMode = DialogBox
 FloatLegStructure = "FRQ:2 CCM:AA"
 Libor = 0.03
 FloatLeg.Init(FloatLegStructure)
 FloatLeg.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_SETTLEMENT, CalcDate
 FloatLeg.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_ISSUE, IssueDate
 FloatLeg.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_MATURITY,
MaturityDate
 FloatLeg.SetAttribute AdxAttrFloatLeg.ATTR1F_FLOATLEG_CURRENT_INDEX, Libor
'Set Swap attributes and parameters
 Swap.ErrorMode = DialogBox
 SwapStructure = "FRQ:2 CCM:AA"
 CalcDate = "10JAN01"
 MaturityDate = "25JUL10"
 IssueDate = "25MAR00"
 FixedRate = 0.0625
 Libor = 0.03
 Swap.Init(SwapStructure)
Swap.SetAttribute AdxAttrFixedLeg.ATTR2F_FIXEDLEG_COUPON, FixedRate
```

```
 Swap.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_SETTLEMENT, CalcDate
 Swap.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_ISSUE, IssueDate
 Swap.SetAttribute AdxAttrInstrument.ADX_ATTR1D_PROD_MATURITY, MaturityDate
 Swap.SetAttribute AdxAttrFloatLeg.ATTR1F_FLOATLEG_CURRENT_INDEX, Libor
‘Set RateModels attributes and parameters
 DiscountRateModel.ErrorMode = DialogBox
 DiscountRateStructure = "RM:YTM RATETYPE:CMP DCB:AA RATEFRQ:1"
 YcStartDate = "10OCT00"
 Yield = 0.0475
 DiscountRateModel.Init (RateStructure)
 DiscountRateModel.SetAttribute AdxAttrRateModel.ADX_ATTR1R_RATEMODEL_ARRAY,
Yield
 IndexRateModel.ErrorMode = DialogBox
 IndexRateStructure = "RM:YTM RATETYPE:CMP DCB:AA RATEFRQ:1"
 YcStartDate = "10OCT00"
 IndexYield = 0.049
 IndexRateModel.Init(RateStructure)
 IndexRateModel.SetAttribute AdxAttrRateModel.ADX_ATTR1R_RATEMODEL_ARRAY,
IndexYield
'Attach the RateModels to the Legs
 FixedLeg.AttachModel ADX_PTR_MODEL_DISCOUNTRATE, DiscountRateModel
 FLoatLeg.AttachModel ADX_PTR_MODEL_DISCOUNTRATE, DiscountRateModel
 FLoatLeg.AttachModel ADX_PTR_MODEL_INDEXRATE, IndexRateModel
'Attach the Legs to the Swaps
 Swap.AttachInstrument ADX_PTR_INSTRUMENT_RECEIVED, FixedLeg
 Swap.AttachInstrument ADX_PTR_INSTRUMENT_PAID, FloatLeg
'Calculation initialization:
'Set calculation formula
 CalcMethod.ErrorMode = DialogBox
 CalcStructure = "CMT:FORM"
 CalcMethod.Init(CalcStructure)
'Ask the formula pricer to compute the calculation attributes of your bond
'Instrument passed to the calculation method
'The AskToCompute method is called for each attribute to calculate
 Swap.AskToCompute AdxAttrSwap.ADX_ATTR3F_SWAP_NPV
 CalcMethod.AttachInstrument Swap
‘Level #3 attributes computation
 CalcMethod.Compute
'Get the computed value
 Npv = Swap.GetAttribute(AdxAttrSwap.ADX_ATTR3F_SWAP_NPV)
```

See also

- "IAdxSwap Interface" on page 220
- "AdxAttrSwap" on page 334

AdxTermStructure

The `AdxTermStructure` component object implements the `IAdxTermStructure` interface that inherits its properties and methods from `IAdxObject`. Use this object to generate a zero coupon yield curve.

VBA sample

```
'This example shows how to use the zero coupon builder object building a zero coupon curve on the bond market:
'Control declaration and initialization
  Dim TermStructureBuilder As AdxTermStructure
  Set TermStructureBuilder = new AdxTermStructure
'Variable declaration
  Dim RateStructure As Variant
  Dim YcStartDate As Variant
  Dim InputArray As Variant
'Variable declaration for result array
  Dim ZcArray As Variant
'Set Zero Coupon builder attributes and parameters
  TermStructureBuilder.ErrorMode = DialogBox
  RateStructure = "RM:YC RATETYPE:CMP DCB:A0"
  YcStartDate = "14JUN01"
  Redim Preserve InputArray (0 To 5, 0 To 5)
'First Column : instrument type
  InputArray (0,0) = "B"
  InputArray (0,1) = "B"
  InputArray (0,2) = "B"
  InputArray (0,3) = "B"
  InputArray (0,4) = "B"
  InputArray (0,5) = "B"
```

```
'Second column : Bonds Start Date
 InputArray (1,0) = "14JUN01"
 InputArray (1,1) = "14JUN01"
 InputArray (1,2) = "14JUN01"
InputArray (1,3) = "14JUN01"
 InputArray (1,4) = "14JUN01"
InputArray (1,5) = "14JUN01"
'Third column : Bonds End Date
 InputArray (2,0) = "21JAN02"
 InputArray (2,1) = "29JAN03"
 InputArray (2,2) = "03JAN05"
 InputArray (2,3) = "04JUL07"
 InputArray (2,4) = "04JAN10"
 InputArray (2,5) = "20JUN16"
'Fourth column : Bonds Coupon
 InputArray (3,0) = 0.08
 InputArray (3,1) = 0.0725
 InputArray (3,2) = 0.07375
 InputArray (3,3) = 0.06
 InputArray (3,4) = 0.05375
 InputArray (3,5) = 0.06
'Fifth column : Bonds market price
 InputArray (4,0) = 1.0270
 InputArray (4,1) = 1.047
 InputArray (4,2) = 1.091
 InputArray (4,3) = 1.067
 InputArray (4,4) = 102.5
 InputArray (4,5) = 1.0644
```

```
'Sixth column : Bonds structure
 InputArray (5,0) = "ACC:AA CLDR:EMU_FI DATED:21JAN1992 FRCD:21JAN1993 FRQ:1
ISSUE:21JAN1992 PX:C PXRND:1E-3:DOWN RP:1 SETTLE:3WD XD:NO PX:C"
 InputArray (5,1) = "ACC:AA CLDR:EMU_FI DATED:29JAN1993 EMC:S FRCD:29JAN1994
FRQ:1  ISSUE:26JAN1993 PX:C PXRND:1E-3:DOWN RP:1 SETTLE:3WD XD:NO PX:C"
 InputArray (5,2) = "ACC:AA CLDR:EMU_FI DATED:03JAN1995 FRCD:03JAN1996 FRQ:1
ISSUE:28DEC1994 PX:C PXRND:1E-3:DOWN RP:1 SETTLE:3WD XD:NO PX:C"
 InputArray (5,3) = "ACC:AA CLDR:EMU_FI DATED:25APR1997 FRCD:04JUL1998 FRQ:1
ISSUE:25APR1997 PX:C PXRND:1E-3:DOWN RP:1 SETTLE:3WD XD:NO PX:C"
 InputArray (5,4) = "ACC:AA CLDR:EMU_FI DATED:22OCT1999 FRCD:04JAN2001 FRQ:1
ISSUE:20OCT1999 PX:C PXRND:1E-3:NEAR RP:1 SETTLE:3WD XD:NO PX:C"
 InputArray (5,5) = "ACC:AA CLDR:EMU_FI DATED:20JUN1986 FRCD:20JUN1987 FRQ:1
ISSUE:20JUN1986 PX:C PXRND:1E-3:DOWN RP:1 SETTLE:3WD XD:NO PX:C"
'Init Builder
 TermStructureBuilder.Init RateStructure
 TermStructureBuilder.SetAttribute  AdxAttrTermStructure.ADX_ATTR1D_
TERMSTRUCTURE_YCSTARTDATE, YcStartDate
TermStructureBuilder.SetInstrumentArray InputArray
'Build the Zero Coupon Curve and get the result
 ZcArray = TermStructureBuilder.BuildTermStructure
```

See also

- "IAdxTermStructure Interface" on page 221
- "AdxAttrTermStructure" on page 337

AdxTimeSeries

The `AdxTimeSeries` component object implements the `IAdxTimeSeries` interface which inherits its properties and methods from "IAdxObject Interface" on page 167.

See also

"IAdxTimeSeries Interface" on page 223

AdxVolatilityModel

The `AdxVolatilityModel` component object implements the `IAdxVolatilityModel` interface that inherits its properties and methods from `IAdxModel`. This object allows you to define a constant volatility curve.

VBA sample

See "AdxOption" on page 275 to see how to use `AdxVolatilityModel`.

See also

- "IAdxVolatilityModel Interface" on page 223
- "AdxAttrVolatilityModel" on page 341

AdfinX Analytics Parameters and Constants

AdfinX Analytics Enumerated Type Overview

The enumerations define sets of attributes used by the different objects. Three types of attributes are specified:

- Attributes that describe the object properties. They are called "Level #1 Attributes" and prefixed `ADX_ATTR1`.
- Calculated attributes. They can be retrieved at the object level. They are called "Level #2 Attributes" and prefixed `ADX_ATTR2`.
- Calculated attributes. They can only be retrieved after computation. They are called "Level #3 Attributes" and prefixed `ADX_ATTR3`.

Attribute types can be known by their names. This is the meaning of the following letters D, E, F, L, R, and S:

- “D" for `CDate`
- “E" for `enumerated`
- “F" for `FLOAT64`
- “L" for `INT32`
- “R" for a range
- “S" for a String

VBA sample

`ADX_ATTR1L_OPT_ASSETCOUNT` This attribute describes a property of the AdxAsset component, e.g. the number of assets, specified as a number by the “L" letter.

Note

The enumerations also define the model and instrument identifiers. They are prefixed `ADX_PTR`.

AdxAttrAsian

The `AdxAttrAsian` enumeration defines the set of attributes used by the `AdxAsian` component. These attributes, which describe the component properties, are called “Level #1 Attributes” and prefixed ADX_ ATTR1. The asset and Asian option identifiers are prefixed ADX_PTR.

Level #1 Attributes

`ADX_ATTR1F_AVERAGE`	Average
`ADX_ATTR1D_FIRST_FIXING`	First fixing date
`ADX_ATTR1E_UND_ASIAN`	Underlying attached to the Asian option
`ADX_ATTR1L_NB_FIXING`	Number of fixings used to calculate the average
`ADX_ATTR1E_UND_AVE`	Average price of the underlying from the first fixing date to the calculation date

Instrument Identifiers

`ADX_PTR_INST_ASSET`	Identifier of the underlying asset

See also

- "AdxAsian" on page 229
- "IAdxAsian Interface" on page 179

AdxAttrAsset

The `AdxAttrAsset` enumeration defines the set of attributes used by the `AdxAsset` component. These attributes, which describe the component properties, are called “Level #1 Attributes” and prefixed ADX_ ATTR1. The calculated attributes that can be retrieved at the object level are called "Level #2 Attribute" and prefixed ATTR2. The model identifiers are prefixed ADX_PTR.

Level #1 Attributes

`ADX_ATTR1E_ASSET_DIV`	Dividend Yield Value

ADX_ATTR1E_ASSET_DIVTYPE	Dividend Type of the Asset: YIELD, FIXED, DISCOUNT, PROPORTIONNAL
ADX_ATTR1E_ASSET_UI	Underlying Instrument: SECURITIES, COMMODITIES, FUTURES, and CURRENCIES
ADX_ATTR1F_ASSET_PRICE	Price of the Underlying Instrument

Level #2 Attributes

ADX_ATTR2F_ASSET_IRR	Internal Interest rate for the Dividend Discount Model
ADX_ATTR2F_ASSET_RISKPREMIUM	Risk premium for the Dividend Discount Model
ADX_ATTR2F_ASSET_THPRICE	Theoretical price for the Dividend Discount Model

Instrument Identifiers

ADX_PTR_MODEL_DIVIDEND	Dividend Model attached to the equity

See also

- "AdxAsset" on page 230
- "IAdxAsset Interface" on page 180

AdxAttrAssetSwap

The AdxAttrAssetSwap enumeration defines the set of attributes used by the AdxAssetSwap component. These attributes, which describe the component properties, are called “Level #1 Attributes" and prefixed ADX_ATTR1. The calculated attributes that can be retrieved at the object level are called "Level #2 Attribute" and prefixed ATTR2. The attributes that can only be retrieved after computation are called “Level #3 Attributes" and prefixed ADX_ATTR3. The model identifiers are prefixed ADX_PTR.

Level #1 Attributes

ADX_ATTR1D_ASSETSWAP_ASRED	
ADX_ATTR1E_ASSETSWAP_ASPX	
ADX_ATTR1E_ASSETSWAP_LEGTYPE	Type of the current leg {FIXED, FLOAT}
ADX_ATTR1F_ASSETSWAP_STRIKE	
ADX_ATTR1S_ASSETSWAP_CROSS	Cross-currency parameter for asset swaps

Level #2 Attributes

ADX_ATTR2R_ASSETSWAP_CASHFLOWS	Cashflows for the asset swap
ADX_ATTR2S_ASSETSWAP_STRUCTURE	Structure of the asset swap

Level #3 Attributes

ADX_ATTR3F_ASSETSWAP_SPREAD	Asset swap spread
ADX_ATTR3R_ASSETSWAP_SPREAD	Asset swap spread

Instrument Identifiers

ADX_PTR_INSTRUMENT_LEG	
ADX_PTR_INSTRUMENT_SWAP_UNDERLYING	

See also

- "AdfinX Analytics Objects" on page 228
- "AdfinX Analytics Interfaces" on page 167

AdxAttrBarrierCapFloor

The AdxAttrBarrierCapFloor enumeration defines the set of attributes used by the AdxBarrierCapFloor component. The attributes that describe the component properties are called “Level #1 Attributes” and prefixed ADX_ATTR1.

Level #1 Attributes

ADX_ATTR1R_CAPFLOOR_REBATE_DOWN	Single barrier or lower rebate
ADX_ATTR1R_CAPFLOOR_REBATE_UP	Upper rebate barrier
ADX_ATTR1R_CAPFLOOR_BARRIER_DOWN	Single barrier or lower barrier
ADX_ATTR1R_CAPFLOOR_BARRIER_UP	Upper barrier

See also

- "AdxBarrierCapFloor" on page 230
- "IAdxBarrierCapFloor Interface" on page 181

AdxAttrBasket

The AdxAttrBasket enumeration defines the set of attributes used by the AdxBasket component. The attributes that describe the component properties are called “Level #1 Attributes” and prefixed ADX_ATTR1. The calculated attributes that can be retrieved at the object level are called "Level #2 Attribute" and prefixed ADX_ATTR2. The attributes that can only be retrieved after computation are called “Level #3 Attributes” and prefixed ADX_ATTR3.

Level #1 Attributes

ADX_ATTR1F_EXO_WEIGHT	Vector of weights
ADX_ATTR1L_OPT_ASSETCOUNT	Number of assets

Level #2 Attributes

ADX_ATTR2F_BSKT_SPOT	Global price
ADX_ATTR2F_BSKT_VOLAT	Global volatility
ADX_ATTR2F_BSKT_YIELD	Global yield

Level #3 Attributes

ADX_ATTR3F_BSKT_SPOT	Global price (deprecated, for backward compatibility, use ADX_ATTR2F_BSKT_SPOT instead)
ADX_ATTR3F_BSKT_VOLAT	Global Volatility (deprecated, for backward compatibility, use ADX_ATTR2F_BSKT_VOLAT instead)
ADX_ATTR3F_BSKT_YIELD	Global yield (deprecated, for backward compatibility, use ADX_ATTR2F_BSKT_YIELD instead)

See also

- "AdxBasket" on page 233
- "IAdxBasket Interface" on page 182

AdxAttrBond

The `AdxAttrBond` enumeration defines the set of attributes used by the `AdxBond` component. The attributes that describe the component properties are called “Level #1 Attributes” and prefixed ADX_ATTR1. The calculated attributes that can be retrieved at the object level are called "Level #2 Attribute" and prefixed ADX_ATTR2. The attributes that can only be retrieved after computation are called “Level #3 Attributes” and prefixed ADX_ATTR3.

Level #1 Attributes

ADX_ATTR1D_BOND_MATYTBONDTEDSPREAD	Maturity of the second Bond used for Ted Spread pricing
ADX_ATTR1E_BOND_TAXCREDIT	Tax credit when TP > RP
ADX_ATTR1E_BOND_TAXPRORATA	Tax on pro rata discount.
ADX_ATTR1F_BOND_AFTERTAX	After tax rate.
ADX_ATTR1F_BOND_COUPONBONDTEDSPREAD	Coupon of the second Bond using for Ted Spread pricing
ADX_ATTR1F_BOND_TAXPRICE	Tax price
ADX_ATTR1R_BOND_TERMSTRUCTURETEDSPREAD	Instrument Array used to compute the Term Structure for Adjusted Ted Spread
ADX_ATTR1S_BOND_STRUCTBONDTEDSPREAD	BondStructure of the second Bond used for Ted Spread Pricing
ADX_ATTR1S_BOND_TAX	Tax on coupon and capital gains
ADX_ATTR1S_BOND_YM	Yield model

Level #2 Attributes

ADX_ATTR2R_BOND_PROCEEDS	Bond Proceeds

Level #3 Attributes

ADX_ATTR3D_BOND_YTWYTBDATE	Bond call or put exercise date (Yield to worst or yield to best)

ADX_ATTR3F_BOND_BASEVOLATILITY	Base volatility
ADX_ATTR3F_BOND_AVGLIFE	Bond average Life
ADX_ATTR3F_BOND_CLEANPRICE	Bond clean price
ADX_ATTR3F_BOND_CONVEXITY	Bond convexity
ADX_ATTR3F_BOND_DURATION	Bond duration
ADX_ATTR3F_BOND_GROSSPRICE	Bond gross price
ADX_ATTR3F_BOND_HEDGING_RATIO	Ted Spread Hedging ratio
ADX_ATTR3F_BOND_IMPYIELDTEDSPREAD	Implied Yield TED (Implied EuroDollar Yield Vs treasury yield)
ADX_ATTR3F_BOND_OPTIONFREEPRICE	Free price of the bond option
ADX_ATTR3F_BOND_PRICE	Bond price
ADX_ATTR3F_BOND_PRICEBONDTEDSPREAD	Price of the second Bond used for Ted Spread pricing
ADX_ATTR3F_BOND_PVBP	Bond pvbp
ADX_ATTR3F_BOND_SPREAD	Bond spread
ADX_ATTR3F_BOND_TEDSPREAD	Ted Spread (EuroDollar Strip vs Treasury Yield)
ADX_ATTR3F_BOND_TEDSPREADADJUST	Adjusted TED Spread (Fixed Spread to Eurodollars)
ADX_ATTR3F_BOND_TPRICE	Theoretic Bond price
ADX_ATTR3F_BOND_VOLATILITY	Bond volatility
ADX_ATTR3F_BOND_YIELD	Bond yield
ADX_ATTR3R_BOND_CASHFLOWDATES	Array of cashflow dates
ADX_ATTR3R_BOND_TEDSPREADFWD	Forward term TED spread
ADX_ATTR3R_BOND_YLDSCHEDULE	Array of yields and dates

See also

- "AdxBond" on page 235
- "IAdxBond Interface" on page 183

AdxAttrCalcMethod

The AdxAttrCalcMethod enumeration defines the set of attributes used by the AdxCalcMethod component. These attributes, which describe the component properties, are called “Level #1 Attributes” and prefixed ADX_ATTR1. The model identifiers are prefixed ADX_PTR.

Level #1 Attributes

ADX_ATTR1E_BND_CSPREAD	SPREAD: credit spread, Yes / No
ADX_ATTR1E_CALC_BSVOL	Volatility used in calculation of Black and Scholes Premium
ADX_ATTR1E_CALC_CMT	Calculation Method Type (TREE, FORM)
ADX_ATTR1E_CALC_CONV	Approximation Type (MIDDLE, RIGHT, CONV) for convexity calculation

ADX_ATTR1F_BND_CSHIFT	SHIFT: credit shift, default value is 0.0001
ADX_ATTR1E_CALC_DCP	Discard First Let (YES,NO) for cap/floor calculation
ADX_ATTR1E_CALC_DUR	Approximation Type (MIDDLE, RIGHT) for duration calculation
ADX_ATTR1E_CALC_FROM	Input value used to solve the specific function. (ex: Implied Volatility)
ADX_ATTR1E_CALC_IGNO	Flag to indicate if options are ignored in convertible bonds calculation (YES, NO)
ADX_ATTR1E_CALC_IRSPVBP	Calculation type for IRS pvbp
ADX_ATTR1E_CALC_PVBP	Approximation Type (MIDDLE, RIGHT) for pvbp calculation
ADX_ATTR1E_CALC_VOL	Approximation Type (MIDDLE, RIGHT) for volatility calculation
ADX_ATTR1E_SOLVER	Volatility Method for impliedvol calculation
ADX_ATTR1F_BND_CSHIFT	SHIFT : credit shift, default value is 0.0001
ADX_ATTR1F_OPT_VOLAT	Volatility value for impliedvol calculation
ADX_PTR_INSTRUMENT_CALC	Instrument attached to the calculation method.

Instrument Identifiers

ADX_PTR_INSTRUMENT_CALC	Instrument attached to the calculation method

See also

- "AdxCalcMethod" on page 238
- "IAdxCalcMethod Interface" on page 172

AdxAttrCapFloor

The AdxAttrCapFloor enumeration defines the set of attributes used by the AdxCapFloor component. The attributes that describe the component properties are called “Level #1 Attributes” and prefixed ADX_ ATTR1. The attributes that can only be retrieved after computation, are called “Level #3 Attributes” and prefixed ADX_ATTR3.

Level #1 Attributes

ADX_ATTR1E_CAPFLOOR_CAPFLOORTYPE	CapFloor Type (Cap, Floor, Collar)
ADX_ATTR1E_CAPFLOOR_CONVBIAS	To calculate forward rates with Convexity Bias
ADX_ATTR1E_CAPFLOOR_FIXING	Fixing Frequency for the forward rate
ADX_ATTR1E_CAPFLOOR_PAYMENT	Payment at beginning or end of caplet period
ADX_ATTR1E_CAPFLOOR_REBATE_ADJ	Rebate is annualized or adjusted to the caplet period
ADX_ATTR1E_CAPFLOOR_RESET	Reset parameter for the forward rate
ADX_ATTR1E_CAPFLOOR_STRGY	Strategy type when pricing bermuda options

ADX_ATTR1F_CAPFLOOR_MCSPREAD	Spread of the Cap/Floor
ADX_ATTR1L_CAPFLOOR_KP	Number of kinky points when pricing bermuda options with MC
ADX_ATTR1L_CAPFLOOR_NBEX	Number of possible exercises when pricing flexi caps/floors
ADX_ATTR1L_CAPFLOOR_PRUNS	Number of primary runs used when pricing bermuda type options with MC
ADX_ATTR1R_CAPFLOOR_STRIKECAP	Array of strike caps
ADX_ATTR1R_CAPFLOOR_STRIKEFLOOR	Array of strike floors

Level #3 Attributes

ADX_ATTR3F_CAPFLOOR_CONV	Convexity of the Cap/Floor
ADX_ATTR3F_CAPFLOOR_IMPSTRIKE	Implied Strike of the Cap/Floor
ADX_ATTR3F_CAPFLOOR_IMPVOL	Implied volatility of the Cap/Floor
ADX_ATTR3F_CAPFLOOR_PREMIUM	Premium of the Cap/Floor
ADX_ATTR3F_CMSSPREADOPTION_IMPVOL1	Implied volatility of the Cap/Floor
ADX_ATTR3F_CMSSPREADOPTION_IMPVOL2	Implied volatility of the Cap/Floor
ADX_ATTR3R_CAPFLOOR_BS_ALLIN	AllIn Cap/Floor Volatility
ADX_ATTR3R_CAPFLOOR_BS_CAPLETVOL	Array of Caplet Dates and Volatilties
ADX_ATTR3R_CAPFLOOR_CAPLETS	Array of caplets
ADX_ATTR3R_CAPFLOOR_DERIV	Array of cap/floor derivatives

See also

- "AdxCapFloor" on page 239
- "IAdxCapFloor Interface" on page 187

AdxAttrCashFlow

The AdxAttrCashFlow enumeration defines the set of attributes used by the AdxCashFlow component. These attributes that can only be retrieved after computation, are called “Level #3 Attributes” and prefixed ADX_ATTR3.

Level #3 Attributes

ADX_ATTR3F_CASHFLOW_AVGLIFE	Average life of a set of cash flows
ADX_ATTR3F_CASHFLOW_CONVEXITY	Convexity of a set of cash flows
ADX_ATTR3F_CASHFLOW_DURATION	Duration of a set of cash flows
ADX_ATTR3F_CASHFLOW_PRICE	Price of a set of cash flows
ADX_ATTR3F_CASHFLOW_PVBP	Pvbp of a set of cash flows
ADX_ATTR3F_CASHFLOW_SPREAD	Spread over a rate model of a set of cash flows
ADX_ATTR3F_CASHFLOW_VOLATILITY	Volatility of a set of cash flows

ADX_ATTR3F_CASHFLOW_YIELD	Yield of a set of cash flows

See also

- "AdxCashFlow" on page 241
- "IAdxCashFlow Interface" on page 188

AdxAttrCDOTranche

The AdxAttrCDOTranche enumeration defines the set of attributes used by the CDOTranche component. The attributes that describe the component properties are called “Level #1 Attributes" and prefixed ADX_ ATTR1. These attributes that can only be retrieved after computation, are called “Level #3 Attributes" and prefixed ADX_ATTR3.

Level #1 Attributes

ADX_ATTR1E_CDO_ATTACH_POINT	The attachment point of the CDO Tranche, expressed as percentage of the aggregate loss
ADX_ATTR1E_CDO_DETACH_POINT	The detachment point of the CDO Tranche, expressed as percentage of the aggregate loss
ADX_ATTR1E_CDO_TRANCHE_NOMINAL	
ADX_ATTR1E_CDO_TYPE	
ADX_ATTR1L_CDO_POOL	
ADX_ATTR1R_CDO_NOMINAL_ARRAY	

Level #3 Attributes

ADX_ATTR3F_CDO_BASE_CORRELATION	Base correlate of the CDO tranche
ADX_ATTR3F_CDO_IMPLIED_CORRELATION	Implied correlate of the CDO tranche
ADX_ATTR3F_CDO_NPV	Net present value of the CDO tranche
ADX_ATTR3F_CDO_PREMIUM	Premium used for the CDO tranche

See also

- "AdfinX Analytics Objects" on page 228

AdxAttrChooser

The AdxAttrChooser enumeration defines the set of attributes used by the AdxChooser component. The model identifiers are prefixed ADX_PTR.

Instrument Identifiers

ADX_PTR_CHOOSER_CALL_OPTION	Call Option for the chooser

ADX_PTR_CHOOSER_PUT_OPTION	Put Option for the chooser

See also

- "AdfinX Analytics Objects" on page 228

AdxAttrConvBond

The AdxAttrConvBond enumeration defines the set of attributes used by the AdxConvBond component. The attributes that describe the component properties are called “Level #1 Attributes” and prefixed ADX_ ATTR1. The calculated attributes that can be retrieved at the object level are called "Level #2 Attribute" and prefixed ADX_ATTR2. The attributes that can only be retrieved after computation are called “Level #3 Attributes” and prefixed ADX_ATTR3. The identifier of the asset attached to the convertible is prefixed ADX_ PTR.

Level #1 Attributes

ADX_ATTR1E_CB_AOC	Accrued payable on conversion
ADX_ATTR1E_CB_IOTYPE	Input and Output format
ADX_ATTR1R_CB_CONVRATIO	Array of periods and values of conversion ratio
ADX_ATTR1R_CB_HURDLE	Array of soft call features
ADX_ATTR1S_CB_CROSS	Cross Currency code

Level #2 Attributes

ADX_ATTR2F_CB_CONV_VALUE	Conversion value of the convertible bond
ADX_ATTR2F_CB_CVPRICE	Conversion price
ADX_ATTR2F_CB_PARITY	Parity

Level #3 Attributes

ADX_ATTR3F_CB_BREAKEVEN	Break even period
ADX_ATTR3F_CB_CONV_VALUE	Conversion value of the convertible bond
ADX_ATTR3F_CB_EQ_PREMIUM	Equity premium
ADX_ATTR3F_CB_IMPLIEDSPREAD	Convertible Bond Implied Spread Calculation
ADX_ATTR3F_CB_IMPLIEDVOL	Implied Volatility for the convertible bond
ADX_ATTR3F_CB_MPREMIUM	Market conversion premium
ADX_ATTR3F_CB_OPT_PREMIUM	Call/Put premium
ADX_ATTR3F_CB_PARITY	Backward compatibility : Parity
ADX_ATTR3F_CB_PREMIUM	Premium of the convertible bond
ADX_ATTR3F_CB_PRICE	Convertible bond price

Identifier

ADX_PTR_INSTRUMENT_ASSET	Asset attached to the convertible bond

ADX_PTR_MODEL_CBFX	Forex model to be attached to the convertible bond

See also

- "AdxConvBond" on page 247
- "IAdxConvBond Interface" on page 190

AdxAttrCorrelation

The AdxAttrCorrelation enumeration defines the set of attributes used by the AdxCorrelation component. This attribute, which describes the component properties, is called “Level #1 Attribute” and prefixed ADX_ATTR1.

Level #1 Attribute

ADX_ATTR1R_COR_MATRIX	Correlation Matrix declaration

See also

- "AdxCorrelation" on page 250
- "IAdxModelBuilder Interface: CreateVolatilityModel" on page 176

AdxAttrCrossCurrency

The AdxAttrCrossCurrency enumeration defines the set of attributes used by the AdxCrossCurrency component. The attributes that describe the component properties are called “Level #1 Attributes” and prefixed ADX_ATTR1. The calculated attributes that can be retrieved at the object level are called "Level #2 Attribute" and prefixed ATTR2. The attributes that can only be retrieved after computation are called “Level #3 Attributes” and prefixed ADX_ATTR3. Instrument identifiers are prefixed ADX_PTR.

Level #1 Attributes

ADX_ATTR1E_CROSS_FROM	Date calculation origin for period calculation
ADX_ATTR1E_CROSS_IGNHOL	Indicator whether the reference currency is taken into account
ADX_ATTR1E_CROSS_IMDEP	Interpolation Method for deposits
ADX_ATTR1E_CROSS_IMSWP	Interpolation Method for swap points
ADX_ATTR1E_CROSS_INVP	Management of invalid period for overnight and tom next
ADX_ATTR1E_CROSS_QM1	Quotation of the first currency versus the base currency when different from USD
ADX_ATTR1E_CROSS_QM2	Quotation of the second currency versus the base currency when different from USD
ADX_ATTR1F_CROSS_QU	Quotation unit
ADX_ATTR1F_CROSS_SWPR	Swap point ratio
ADX_ATTR1L_CROSS_CRDEC	Cross rate decimals
ADX_ATTR1L_CROSS_PERIOD_DAYS	Number of days in the cross period

ADX_ATTR1L_CROSS_PERIOD_OFFSET	Offset between calcdate and cross date
ADX_ATTR1L_CROSS_SPDEC	Swap point decimals
ADX_ATTR1R_CROSS_PERIODARRAY	Array of period/dates asked in computation
ADX_ATTR1S_CROSS_PERIODVALUE	Period code of crosscurrency

Level #2 Attributes

ADX_ATTR2R_CALC_PERIOD	Calculates period dates in FxCalcPeriod() order
ADX_ATTR2S_CROSS_PERIODDATES	Calculates period dates (deprecated)

Level #3 Attributes

ADX_ATTR3D_CROSS_SPOT1_DATE	Spot value date of the Currency1
ADX_ATTR3D_CROSS_SPOT12_DATE	Spot value date of the cross currency
ADX_ATTR3D_CROSS_SPOT2_DATE	Spot value date of the Currency2
ADX_ATTR3F_CROSS_IMPLIEDDEPOSIT_ASK	Ask of the cross swap point
ADX_ATTR3F_CROSS_IMPLIEDDEPOSIT_BID	Bid of the cross swap point
ADX_ATTR3F_CROSS_SPOT12_ASK	Ask side of the spot cross rate
ADX_ATTR3F_CROSS_SPOT12_BID	Bid side of the spot cross rate
ADX_ATTR3F_CROSS_SWPPT12_ASK	Ask side of the cross swap point
ADX_ATTR3F_CROSS_SWPPT12_BID	Bid side of the cross swap point
ADX_ATTR3F_FX_GENCALC	Performs any kind of Forex or Money market calculation from an instrument code
ADX_ATTR3L_CROSS_END_DATE	End date of the period
ADX_ATTR3L_CROSS_START_DATE	Start date of the period
ADX_ATTR3R_CROSS_CROSSSPOT	Calculates the spot cross rate, cross value date and the spot dates being equal
ADX_ATTR3R_CROSS_DEPTOSWP	Calculates the synthetic swap point from deposit using arrays of deposit
ADX_ATTR3R_CROSS_SWPTODEP_DEP1	Calculates the synthetic deposit from swap point using an array of swap point and deposit

ADX_ATTR3R_CROSS_SWPTODEP_DEP2	Calculates the synthetic deposit from swap point using an array of swap point and deposit
ADX_ATTR3R_CROSS_SWPTODEP_DEPCUR	Calculates the synthetic deposit from swap point using an array of swap point and deposit
ADX_ATTR3R_CROSS_SWPTODEP_DEPUSD	Calculates the synthetic deposit from swap point using an array of swap point and deposi
ADX_ATTR3R_CROSS_SWPTOSWP	Calculates the swap points of cross given swap point of currency

Identifiers

ADX_PTR_INSTRUMENT_CUR_PIVOT	Identifier of the currency pivot
ADX_PTR_INSTRUMENT_CUR1	Identifier of the first currency
ADX_PTR_INSTRUMENT_CUR2	Identifier of the second currency

See also

- "AdxCrossCurrency" on page 251
- "IAdxCrossCurrency Interface" on page 192

AdxAttrCurrency

The AdxAttrCurrency enumeration defines the set of attributes used by the AdxCurrency component. The attributes that describe the component properties are called “Level #1 Attributes” and prefixed ADX_ ATTR1. The attributes that can only be retrieved after computation are called “Level #3 Attributes” and prefixed ADX_ATTR3. Model identifiers are prefixed ADX_PTR.

Level #1 Attributes

ADX_ATTR1E_CUR_FROM	Date calculation origin for period calculation
ADX_ATTR1E_CUR_STYLE_DTM	Number of working days from trade date to MM spot date
ADX_ATTR1E_CUR_STYLE_DTS	Number of working days from trade date to Forex spot date
ADX_ATTR1E_CUR_STYLE_QM	Quotation mode
ADX_ATTR1F_CUR_SPOT_ASK	Ask spot rate of the currency
ADX_ATTR1F_CUR_SPOT_BID	Bid spot rate of the currency
ADX_ATTR1F_CUR_SWPPT_ASK	Ask swap point of the currency
ADX_ATTR1F_CUR_SWPPT_BID	Bid swap point of the currency
ADX_ATTR1L_CUR_PERIOD_DAYS	Number of days in the period
ADX_ATTR1L_CUR_PERIOD_OFFSET	Offset between currency spot date and cross spot date
ADX_ATTR1L_CUR_STYLE_YB	Money Market year basis
ADX_ATTR1R_CUR_DEPOSIT_BIDASK	Array of deposit (period, bid, ask)
ADX_ATTR1R_CUR_SWPPT_BIDASK	Array of swap point (period, bid, ask)
ADX_ATTR1S_CUR_STYLE_CODE	Currency style code
ADX_ATTR1S_CUR_STYLE_NAME	Currency style name

Level #3 Attributes

ADX_ATTR3F_CUR_SWPPT_ASK	Ask swap point of the currency (deprecated, for backward compatibility, use ADX_ATTR1F_CUR_SWPPT_ASK instead)
ADX_ATTR3F_CUR_SWPPT_BID	Bid swap point of the currency (deprecated, for backward compatibility, use ADX_ATTR1F_CUR_SWPPT_BID instead)
ADX_ATTR3L_CUR_PERIOD_DAYS	Number of days in the period (deprecated, for backward compatibility, use ADX_ATTR1L_CUR_PERIOD_DAYS instead)
ADX_ATTR3L_CUR_PERIOD_OFFSET	Offset between currency spot date and cross spot date (deprecated, for backward compatibility, use ADX_ATTR1L_CUR_PERIOD_OFFSET instead)

Identifiers

ADX_PTR_MODEL_DISCOUNTRATE_BID	Discount rate model attached to the Bid of the Currency
ADX_PTR_MODEL_DISCOUNTRATE_ASK	Discount rate model attached to the Ask of the Currency

See also

- "AdxCurrency" on page 252
- "IAdxCurrency Interface" on page 193

AdxAttrDigitalCapFloor

The AdxAttrDigitalCapFloor enumeration defines the set of attributes used by the AdxDigitalCapFloor component. These attributes, which describe the component properties, are called “Level #1 Attributes” and prefixed ADX_ATTR1.

Level #1 Attributes

ADX_ATTR1R_DIGITAL_CAP_REBATE	Rebate value for a digital cap
ADX_ATTR1R_DIGITAL_FLOOR_REBATE	Rebate value for a digital floor

See also

- "AdxDigitalCapFloor" on page 254
- "IAdxDigitalCapFloor Interface" on page 194

AdxAttrDividendModel

The AdxAttrDividendModel enumeration defines the set of attributes used by the AdxDividendModel component. These attributes, which describe the component properties, are called “Level #1 Attributes” and prefixed ADX_ATTR1.

Level #1 Attributes

ADX_ATTR1E_ASSET_DCB	Day Count Basis Value
ADX_ATTR1E_ASSET_RATETYPE	Rate Type : ACTUAL, CONTINUOUS, MONEYMARKET, DISCOUNT

ADX_ATTR1E_DIVIDEND_CSTGROWTH_TYPE	Dividend Growth: Historical or Estimated
ADX_ATTR1E_DIVIDEND_FRQ	Compounding frequency
ADX_ATTR1E_DIVIDEND_TYPE	Dividend Type: CONT, DISC, FIXED, PROP
ADX_ATTR1E_DIVIDEND_USERDEFINED	User Defined Dividend (optional)
ADX_ATTR1R_DIVIDEND	Dividend Yield Value
ADX_ATTR1R_DIVIDEND_GROWTH	Array of dividend growth

See also

- "AdxDividendModel" on page 257
- "IAdxDividendModel Interface" on page 195

AdxAttrFiniteDiff

The AdxAttrFiniteDiff enumeration defines the set of attributes used by the AdxFiniteDiff component. The attributes that describe the component properties are called “Level #1 Attributes” and prefixed ADX_ATTR1.

Level #1 Attributes

ADX_ATTR1E_FD_NBFACTOR	Number of factors for the finite difference method
ADX_ATTR1L_FD_TITER	Number of time steps for the finite difference method
ADX_ATTRE_FD_NBFACTOR	Number of factors for the finite difference method (deprecated, for backward compatibility, use ADX_ATTR1E_FD_NBFACTOR instead)
ADX_ATTRL_FD_TITER	Number of time steps for the finite difference method (deprecated, for backward compatibility, use ADX_ATTR1L_FD_TITER instead)

See also

- "IAdxCalcMethod Interface" on page 172

AdxAttrFixedLeg

The AdxAttrFixedLeg enumeration defines the set of attributes used by the AdxFixedLeg component. The attribute that describes the component properties is called “Level #1 Attribute” and prefixed ADX_ATTR1. The calculated attributes that can be retrieved at the object level are called “Level #2 Attribute” and prefixed ADX_ATTR2.

Level #1 Attribute

ADX_ATTR1F_FIXEDLEG_COUPON	Nominal coupon rate to be applied to the fixed leg
ADX_ATTR1F_FIXEDLEG_STEP	Step value to be applied to the fixed leg

Level #2 Attributes

ADX_ATTR2F_FIXEDLEG_COUPON	Backward compatibility: Nominal coupon rate to be applied to the fixed leg
ADX_ATTR2O_FIXEDLEG_CASHFLOWS	Backward compatibility: Fixed leg cash flows

See also

- "AdxFixedLeg" on page 257
- "IAdxFixedLeg Interface" on page 196

AdxAttrFloatLeg

The AdxAttrFloatLeg enumeration defines the set of attributes used by the AdxFloatLeg component. The attributes that describe the component properties are called “Level #1 Attributes” and prefixed ADX_ATTR1. The calculated attribute that can be retrieved at the object level is called “Level #2 Attribute” and prefixed ADX_ATTR2. The model identifier is prefixed ADX_PTR.

Level #1 Attributes

ADX_ATTR1S_FLOATLEG_AOD	Flag to indicate accrued payment on default (deprecated, for backward compatibility, use ADX_ATTR1E_LEG_AOD instead)
ADX_ATTR1E_FLOATLEG_CPLAG	Backward compatibility: Contingent payment lag
ADX_ATTR1E_FLOATLEG_INDEXDATE	Input date to express dates in start dates or end dates of coupons
ADX_ATTR1E_FLOATLEG_RESETFREQUENCY	Reset Frequency: Frequency of the Reference Rate Calculations
ADX_ATTR1E_LEG_AOD	Flag to indicate accrued payment on default
ADX_ATTR1F_FLOATLEG_CAP	Cap Value
ADX_ATTR1F_FLOATLEG_COLLAR	Collar Value
ADX_ATTR1F_FLOATLEG_FLOOR	Floor Value
ADX_ATTR1F_FLOATLEG_SPREAD	Spread to be applied to the index
ADX_ATTR1L_FLOATLEG_FIXINGDELAY	For Chinese IRS: Delay between the calendar repo and the fixing date
ADX_ATTR1R_FLOATLEG_CURRENT_INDEX	Current index Array
ADX_ATTR1R_FLOATLEG_INDEX_SCENARIO	Index scenario Array for cash flows
ADX_ATTR1R_FLOATLEG_PROJECTED_INDEX	Projected index Array

ADX_ATTR1S_CMS_TENOR	Cms Tenor
ADX_ATTR1S_FLOATLEG_AOD	Backward compatibility : Flag to indicate accrued payment on default (use ADX_ATTR1S_LEG_AOD instead)
ADX_ATTR1S_FLOATLEG_CPLAG	Contingent payment lag
ADX_ATTR1S_LEG_AOD	Backward compatibility : Flag to indicate accrued payment on default

Level #2 Attribute

ADX_ATTR2O_FLOATLEG_CASHFLOWS	Backward compatibility : Float leg cash flows

Identifier

ADX_PTR_MODEL_INDEXRATE	Index rate model to be attached

See also

- "AdxFloatLeg" on page 257
- "IAdxFloatLeg Interface" on page 197

AdxAttrForex

The AdxAttrForex enumeration defines the set of attributes used by the AdxForex component. The attributes that describe the component properties are called “Level #1 Attributes” and prefixed ADX_ATTR1. The calculated attributes that can be retrieved at the object level are called "Level #2 Attribute" and prefixed ADX_ATTR2.These attributes, which can only be retrieved after computation are called “Level #3 Attributes” and prefixed ADX_ATTR3.

Level #1 Attributes

ADX_ATTR1E_CROSS_QM	Backward compatibility : Quotation mode versus the USD
ADX_ATTR1R_DEP1BA_ARRAY	Cur1: array of dates and deposit rates
ADX_ATTR1R_DEP2BA_ARRAY	Cur2: array of dates and deposit rates
ADX_ATTR1R_DEPUSD_ARRAY	USD: array of dates and deposit rates
ADX_ATTR1R_SWP1BA_ARRAY	Cur1: array of dates and swap points
ADX_ATTR1R_SWP2BA_ARRAY	Cur2: array of dates and swap points
ADX_ATTR1S_FOREX_DIRECTFRMOIDN	Flag: Is forex direct from IDN source
ADX_ATTR1S_FOREX_INSTRUMENTCODE	Code of the instrument

Level #2 Attributes

ADX_ATTR2D_CROSS_PERIOD_ENDDATE_ADJ	Backward compatibility : Period Adjusted End Date
ADX_ATTR2D_CROSS_PERIOD_ENDDATE_NOADJ	Backward compatibility : Period Unadjusted End Date

ADX_ATTR2D_CROSS_PERIOD_STARTDATE	Backward compatibility : Period Start Date
ADX_ATTR2L_CROSS_PERIOD_OFFSET	Backward compatibility : Offset between calcdate and cross date
ADX_ATTR2L_FOREX_INSTRUMENTID	Instrument id from instrument code
ADX_ATTR2L_FX_INSTRUMENTID	Instrument id from instrument code
ADX_ATTR2S_CROSS_PERIOD_MESSAGE	Backward compatibility : Information message on rule followed to adjust period End Date
ADX_ATTR2S_CROSS_PERIOD1	Backward compatibility : First period of forex instrument
ADX_ATTR2S_CROSS_PERIOD2	Second period of forex instrument
ADX_ATTR2S_FOREX_DEV1	Code of the Devise 1
ADX_ATTR2S_FOREX_DEV2	Code of the Devise 2
ADX_ATTR2S_FOREX_DEVVERSUS	Currency code for deposit calculation
ADX_ATTR2S_FOREX_PERIOD1	Code of the first period
ADX_ATTR2S_FOREX_PERIOD2	Code of the second period

Level #3 Attributes

ADX_ATTR3D_CROSS_END1_DATE	Backward compatibility: End date of period
ADX_ATTR3D_CROSS_END2_DATE	End date of forward period
ADX_ATTR3F_CUR_DEPOSIT_ASK	Backward compatibility: Ask of the currency deposit rate
ADX_ATTR3F_CUR_DEPOSIT_BID	Backward compatibility: Bid of the currency deposit rate
ADX_ATTR3F_FOREX_DIRECTFRMOIDN	Flag: Is forex direct from IDN source (deprecated, used for backward compatibility use instead ADX_ATTR1S_FOREX_DIRECTFRMOIDN)
ADX_ATTR3F_FOREX_INSTRUMENTCODE	Code of the instrument (deprecated, used for backward compatibility use instead ADX_ATTR1S_FOREX_INSTRUMENTCODE)
ADX_ATTR3F_FX_CROSS	Backward compatibility: Calculates the spot cross rate, cross value date and the spot dates being equal
ADX_ATTR3F_FX_CROSSA	Backward compatibility: Calculates the adjusted spot cross rate
ADX_ATTR3F_FX_CROSSD	Backward compatibility: Calculates the spot cross rate, cross value date and the spot dates being different
ADX_ATTR3F_FX_DEPTOSWPD	Backward compatibility: Calculates the synthetic swap point from deposit using a number of days
ADX_ATTR3F_FX_DEPTOSWPP	Backward compatibility: Calculates the synthetic swap point from deposit using a period

ADX_ATTR3F_FX_PERIOD	Backward compatibility: Calculates period dates
ADX_ATTR3F_FX_SWPTODEPD_DEP1	Backward compatibility: Calculates the synthetic deposit from swap point using a number of days
ADX_ATTR3F_FX_SWPTODEPD_DEP2	Backward compatibility: Calculates the synthetic deposit from swap point using a number of days
ADX_ATTR3F_FX_SWPTODEPD_DEPCUR	Backward compatibility: Calculates the synthetic deposit from swap point using a number of days
ADX_ATTR3F_FX_SWPTODEPD_DEPUSD	Backward compatibility: Calculates the synthetic deposit from swap point using a number of days
ADX_ATTR3F_FX_SWPTODEPP_DEP1	Backward compatibility: Calculates the synthetic deposit from swap point using a period
ADX_ATTR3F_FX_SWPTODEPP_DEP2	Backward compatibility: Calculates the synthetic deposit from swap point using a period
ADX_ATTR3F_FX_SWPTODEPP_DEPCUR	Backward compatibility: Calculates the synthetic deposit from swap point using a period
ADX_ATTR3F_FX_SWPTODEPP_DEPUSD	Backward compatibility: Calculates the synthetic deposit from swap point using a period
ADX_ATTR3F_FX_SWPTOSWP	Backward compatibility: Calculates the cross swap point from swap points, cross value date and spot dates being equal
ADX_ATTR3F_FX_SWPTOSWPD	Backward compatibility: Calculates the cross swap point from swap points, cross value date and spot dates being different
ADX_ATTR3F_FX_SWPTOSWPP	Backward compatibility: Calculates the cross swap point from swap points, using a period for calculations
ADX_ATTR3L_CROSS_PERIOD_DAYS	Backward compatibility: Number of days in the cross period
ADX_ATTR3S_FOREX_ERRORCODE	Error message

See also

- "AdxForex" on page 258
- "IAdxForex Interface" on page 197

AdxAttrFormula

The `AdxAttrFormula` enumeration defines the set of attributes used by the `AdxCalcMethod` component. These attributes, which describe the component properties, are called “Level #1 Attributes” and prefixed ADX_ATTR1.

Level #1 Attributes

ADX_ATTR1E_BND_CM	Calculation Method

ADX_ATTR1E_BND_EY	Rate Frequency : 1/2/4/12
ADX_ATTR1E_FORMULA_BWB	
ADX_ATTR1E_FORMULA_RETA	Total Return approach
ADX_ATTR1E_FXFORMULA_BA_CTRL	Bid/Ask Control
ADX_ATTR1E_FXFORMULA_DRF	Rate Percent Yes/No
ADX_ATTR1E_FXFORMULA_IGNR	Ignore Quotation Unit and Swap PointRatio Yes/No
ADX_ATTR1E_OPT_FT	Option Formula Type (BS, Whaley, CEV)
ADX_ATTR1E_OPT_MATCORRECTED	Use of Maturity Correction
ADX_ATTR1E_OPT_NORMAL	Type of Normal Cumulative (10-3, 10-6, Hull)
ADX_ATTR1F_FORMULA_SHIFT	
ADX_ATTR1F_FXFORMULA_DEC	Decimal
ADX_ATTR1L_FORMULA_OFFSET	
ADX_ATTR1L_FORMULA_SPREAD	
ADX_ATTR1S_SWAP_DC	Decimal
ADX_ATTRE_BND_BF	Bond Formula Type (TRE)
ADX_ATTRE_BND_CF	Cash-Flow Discounting Method
ADX_ATTRE_BND_CMP	Compounding Frequency
ADX_ATTRE_BND_GC	Clean Price / Gross Price
ADX_ATTRE_BND_LLP	Linear Last Period
ADX_ATTRE_BND_YTM	Deactivate Call or Put schedule
ADX_ATTRE_FXFORMULA_BA_CTRL	Bid/Ask Control (deprecated, for backward compatibility, use ADX_ATTR1E_FXFORMULA_BA_CTRL instead)
ADX_ATTRE_FXFORMULA_DRF	Rate Percent Yes/No (deprecated, for backward compatibility, use ADX_ATTR1E_FXFORMULA_DRF instead)

ADX_ATTRE_FXFORMULA_IGNR	Ignore Quotation Unit and Swap PointRatio Yes/No (deprecated, for backward compatibility, use ADX_ATTR1E_FXFORMULA_IGNR instead)
ADX_ATTRE_OPT_FT	Option Formula Type (BS, Whaley, CEV, Brigo) (deprecated, for backward compatibility, use ADX_ATTR1E_OPT_FT instead)
ADX_ATTRE_OPT_MATCORRECTED	Use of Maturity Correction (deprecated, for backward compatibility, use ADX_ATTR1E_OPT_MATCORRECTED instead)
ADX_ATTRE_OPT_NORMAL	Type of Normal Cumulative (10-3, 10-6, Hull) (deprecated, for backward compatibility, use ADX_ATTR1E_OPT_NORMAL instead)
ADX_ATTRF_FXFORMULA_DEC	Decimal(deprecated, for backward compatibility, use ADX_ATTR1F_FXFORMULA_DEC instead)

See also

- "IAdxCalcMethod Interface" on page 172

AdxAttrFra

The `AdxAttrFra` enumeration defines the set of attributes used by the `AdxFra` component. The attributes that describe the component properties are called “Level #1 Attributes” and prefixed ADX_ATTR1. The attribute that can only be retrieved after computation is called “Level #3 Attributes” and prefixed ADX_ATTR3. The model identifier is prefixed ADX_PTR.

Level #1 Attributes

ADX_ATTR1D_FRA_FIXING	Fixing Date
ADX_ATTR1E_FRA_REFDATE	Contract reference date
ADX_ATTR1L_FRA_YBASIS	Money Market Year Basis
ADX_ATTR1S_FRA_NFVP	For broken FRAs, to use the next period if the maturity date falls in a period of i days before the next period date
ADX_ATTR1S_FRA_PERIOD	FRA maturity date expressed as a code such as "1Y"
ADX_ATTR1S_FRA_PFVP	For broken FRAs, to use the previous period if the maturity date falls in a period of i days after the previous period date

Level #3 Attributes

ADX_ATTR3F_FRA_FORWARDRATE	Forward Rate
ADX_ATTR3F_FRA_FORWARDRATEASK	Forward Rate Ask

Identifier

ADX_PTR_MODEL_FRA_RATE_ASK	Discount rate model attached to the Ask of the FRA

See also

- "AdxFra" on page 258
- "IAdxFra Interface" on page 198

AdxAttrFrn

The `AdxAttrFrn` enumeration defines the set of attributes used by the `AdxFrn` component. The attributes that describe the component properties are called “Level #1 Attributes” and prefixed ADX_ATTR1. The attributes that can only be retrieved after computation are called “Level #3 Attributes” and prefixed ADX_ATTR3.

Level #1 Attributes

ADX_ATTR1E_FRN_YTM_AUTO	Yield type Auto for callable/puttable FRN
ADX_ATTR1E_FRN_YTM_BEST	Yield type Best for callable/puttable FRN
ADX_ATTR1E_FRN_YTM_MATURITY	Yield type Maturity for callable/puttable FRN
ADX_ATTR1E_FRN_YTM_WORST	Yield type Worst for callable/puttable FRN
ADX_ATTR1F_FRN_QUOTED_MARGIN	Quoted Margin of the FRN
ADX_ATTR1F_FRN_REPO_RATE	Carrying repo rate
ADX_ATTR1S_PROD_IQM	Instrument Quotation Mode for Brazil FRN LFT

Level #3 Attributes

ADX_ATTR3D_FRN_YTWYTBDATE	Yield to Worst/Yield to Best Date
ADX_ATTR3F_FRN_ADJUSTED_PRICE	Adjusted Price of the FRN
ADX_ATTR3F_FRN_ADJUSTED_SIMPLE_MARGIN	Adjusted Simple Margin of the FRN
ADX_ATTR3F_FRN_ADJUSTED_TOTAL_MARGIN	Adjusted Total Margin of the FRN
ADX_ATTR3F_FRN_AVGLIFE	Average life of the FRN
ADX_ATTR3F_FRN_BONDES_MARGIN	Calculates the margin of a Bonds FRN
ADX_ATTR3F_FRN_CLEAN_PRICE	Clean Price of the FRN
ADX_ATTR3F_FRN_CONVEXITY	Convexity of the FRN
ADX_ATTR3F_FRN_DISCOUNTED_MARGIN	Discounted Margin of the FRN
ADX_ATTR3F_FRN_DURATION	Duration of the FRN
ADX_ATTR3F_FRN_GROSS_PRICE	Gross Price of the FRN
ADX_ATTR3F_FRN_INDEXDUR	Calculates the index duration
ADX_ATTR3F_FRN_OPTIONFREEPRICE	Option free price
ADX_ATTR3F_FRN_PRICE	Calculates the price
ADX_ATTR3F_FRN_PVBP	Calculates the Price variation per basis point of a FRN
ADX_ATTR3F_FRN_SIMPLE_MARGIN	Simple Margin of the FRN
ADX_ATTR3F_FRN_SPREADDUR	Calculates the spread duration
ADX_ATTR3F_FRN_VOLATILITY	Volatility of the FRN

ADX_ATTR3F_FRN_YIELD	Yield of the FRN
ADX_ATTR3F_FRN_YIELD_SPREAD	Spread of the FRN
ADX_ATTR3O_REPO_REPO_RATE	Backward compatibility: Repo Rate

See also

- "AdxFrn" on page 260
- "IAdxFrn Interface" on page 199

AdxAttrFuture

The AdxAttrFuture enumeration defines the set of attributes used by the AdxFuture component. The attributes that describe the component properties are called “Level #1 Attributes” and prefixed ADX_ATTR1. The attributes that can only be retrieved after computation are called “Level #3 Attributes” and prefixed ADX_ATTR3. The instrument identifier is prefixed ADX_PTR.

Level #1 Attributes

ADX_ATTR1E_FUTURE_DEC	Decimal precision for conversion factor {NO, RND:i, TRUNC:i with i as integer}
ADX_ATTR1E_FUTURE_DTM	NbDays after the 3rd Wed of the month To calculate the contract dates
ADX_ATTR1E_FUTURE_MDADJ	Maturity Adjustment convention
ADX_ATTR1E_FUTURE_QUOT	Quotation mode {100, 32, 256}
ADX_ATTR1E_FUTURE_REFDATE	Contract reference rule for calculation
ADX_ATTR1E_FUTURE_YB	Year basis of the underlying deposit when applicable (STIR Futures)
ADX_ATTR1E_ONFUTURE_AVG	Average
ADX_ATTR1E_ONFUTURE_EMC	End of Month Convention
ADX_ATTR1E_ONFUTURE_REFDATE	Contract reference rule
ADX_ATTR1E_ONFUTURE_YB	Year Basis
ADX_ATTR1F_FUTURE_SIZE	Contract size {i with i as numeric}
ADX_ATTR1F_FUTURE_TICK	Tick value
ADX_ATTR1L_FUTURE_NBCODES	Number of Future Codes to compute
ADX_ATTR1I_FUTURE_NBMC	Number of monthly contracts {i with i as integer}
ADX_ATTR1I_FUTURE_NBQC	Number of quarterly contracts {i with i as integer}
ADX_ATTR1L_FUTURE_ODD	Skipping or not odd contract codes in Future Codes
ADX_ATTR1L_FUTURE_ROLL	Skipping or not the next contract code in Future Codes
ADX_ATTR1S_FUTURE_BOND	Underlying bond style
ADX_ATTR1S_FUTURE_CDADJS	Conversion factor date adjustment {C{:M, Q}, F{:M, Q}, N, P{:M, Q}}
ADX_ATTR1S_FUTURE_CFD	Conversion factor date calculation method {iWD with i as integer}
ADX_ATTR1S_FUTURE_CLDR	Calendar for holiday management {calendar}

ADX_ATTR1S_FUTURE_EDD	Delivery period end date calculation method {LAST, iWD with i as integer}
ADX_ATTR1S_FUTURE_MATURITYCODE	Maturity Code (e.g. : 'H02')
ADX_ATTR1S_FUTURE_NAME	Contract code
ADX_ATTR1S_FUTURE_PERIOD	Contract period length (e.g. : '3M' -> 3 Months)
ADX_ATTR1S_FUTURE_RRTYPE	RateModel type
ADX_ATTR1S_FUTURE_SDD	Delivery period start date calculation method {FIRST, iWD with i as integer}
ADX_ATTR1S_ONFUTURE_IDX	Underlying Index

Level #3 Attributes

ADX_ATTR3D_FUTURE_ENDDATE	End Date of the future contract
ADX_ATTR3D_FUTURE_STARTDATE	Start Date of the future contract
ADX_ATTR3F_FUTURE_CONVFACTOR	Conversion factor
ADX_ATTR3F_FUTURE_RATE	Future Contract rate
ADX_ATTR3R_FUTURE_CODES	Array of Future Codes after the calculation date

Instrument identifier

ADX_PTR_INSTRUMENT_FUT_UNDERLYING	Underlying attached to the Future

See also

- "AdxFuture" on page 263
- "IAdxFuture Interface" on page 201

AdxAttrFxFormula

The `AdxAttrFxFormula` enumeration defines the set of attributes used by the `AdxCalcMethod` component. This attribute, which describes the component properties, is called “Level #1 Attribute” and prefixed ADX_ATTR1.

Level #1 Attribute

ADX_ATTR1F_FXFORMULA_DEC	Decimal

See also

- "IAdxCalcMethod Interface" on page 172

AdxAttrFxModel

The `AdxAttrFxModel` enumeration defines the set of attributes used by the `AdxFxModel` component. These attributes, which describe the component properties, are called “Level #1 Attributes”, and are prefixed ADX_ATTR1.

Level #1 Attributes

ADX_ATTR1D_FXMODEL_STARTDATE	Start Date of the model
ADX_ATTR1E_FXMODEL_DCB	Day Count Basis
ADX_ATTR1E_FXMODEL_IM	Interpolation Method
ADX_ATTR1E_FXMODEL_OBC	Extrapolation Mode
ADX_ATTR1F_FXMODEL_FXSPOTRATE	Fx Spot Rate
ADX_ATTR1R_FXMODEL_ARRAY	Rate Model Array

See also

- "AdxFxModel" on page 266
- "IAdxFxModel Interface" on page 202

AdxAttrIdx

The attributes that describe the component properties are called “Level #1 Attributes" and prefixed ADX_ ATTR1.

Level #1 Attributes

ADX_ATTR1E_IDX_AVG	Index compounding method {ARI, CMP, MCA, NONE}
ADX_ATTR1E_IDX_ PERIOD	Index calculation period {ON, TN}
ADX_ATTR1E_IDX_RND	Rounding decimals for the compounded settlement rate calculation {0, 1, 2, 3, 4, 5, 6}
ADX_ATTR1E_IDX_YB	Money market year basis {360, 365}
ADX_ATTR1F_IDX_RND	Return as FLOAT64: Rounding decimals for the compounded settlement rate calculation
ADX_ATTR1F_IDX_YB	Return as FLOAT64: Money market year basis
ADX_ATTR1S_IDX_ HISTCODE	History Code in Database
ADX_ATTR1S_IDX_RIC	Thomson Reuters Instrument Code (RIC) for the index {RIC name}

AdxAttrIlb

The AdxAttrIlb enumeration defines the set of attributes used by the AdxIlb component. The attributes that describe the component properties are called “Level #1 Attributes” and are prefixed ADX_ATTR1. The calculated attributes that can be retrieved at the object level are called "Level #2 Attribute" and prefixed ADX_ ATTR2. The attributes that can only be retrieved after computation are called “Level #3 Attributes” and prefixed ADX_ATTR3.

Level #1 Attributes

ADX_ATTR1D_ILB_INFLATION_REF_DATE	Inflation Reference Date

ADX_ATTR1E_ILB_ICF	Inflation adjustment method
ADX_ATTR1E_ILB_ICM	Daily inflation reference and coupon calculation method
ADX_ATTR1E_ILB_PRG	Par redemption guarantee
ADX_ATTR1E_LEG_IRB	Ilb inflation rate basis
ADX_ATTR1F_ILB_BRI	Base reference index
ADX_ATTR1F_ILB_IRND	Index rounding tick size
ADX_ATTR1L_ILB_LBM	Number of lookback months
ADX_ATTR1R_ILB_INFLATION_RATEARRAY	Array of anticipated inflation rates
ADX_ATTR1S_ILB_ICF	Backward compatibility: Inflation adjustment method
ADX_ATTR1S_ILB_IDX	Underlying index style

Level #2 Attributes

ADX_ATTR2F_ILB_INDEXRATIO	Ilb Index Ratio

See also

- "AdxIlb" on page 266
- "IAdxIlb Interface" on page 204

AdxAttrInstrument

The AdxAttrInstrument enumeration defines the set of attributes used by each of the interfaces, which derive from the IAdxInstrument interface. The attributes that describe the component properties are called “Level #1 Attributes” and prefixed ADX_ATTR1. The calculated attributes that can be retrieved at the object level are called "Level #2 Attribute" and prefixed ADX_ATTR2. The attribute that can only be retrieved after computation is called “Level #3 Attributes” and prefixed ADX_ATTR3. The model identifier is prefixed ADX_PTR.

Level #1 Attributes

ADX_ATTR1D_PROD_ISSUE	Issue Date
ADX_ATTR1D_PROD_MATURITY	Maturity Date
ADX_ATTR1D_PROD_SETTLEMENT	Settlement Date
ADX_ATTR1D_PROD_TRADE	Trade Date
ADX_ATTR1F_PROD_NOMINAL_AMOUNT	Nominal amount in cash of the instrument
ADX_ATTR1S_PROD_CLDR	Calendar for the product
ADX_ATTR1S_PROD_CUR	Currency code of the instrument
ADX_ATTR1S_PROD_MATURITY_CODE	Maturity Code or Maturity Date of Instrument
ADX_ATTR1S_PROD_SETTLE	Settlement rule

Level #2 Attributes

ADX_ATTR2S_OBJECT_STRUCTURE	Structure of an object

Level #3 Attributes

ADX_ATTR3F_PROD_BUY_PRICE	Instrument buy price in carry operation
ADX_ATTR3F_PROD_GROSS_P_AND_L	Gross for a carry operation
ADX_ATTR3F_PROD_SELL_PRICE	Instrument sell price in carry operation
ADX_ATTR3F_PROD_TOTAL_RETURN	Total return for a carry operation
ADX_ATTR3R_BASKET_CROSSGREEK	Greeks results

Model Identifier

ADX_PTR_CMSPAID_MODEL_VOLATILITY	Attach a volatility to the paid cms leg
ADX_PTR_CMSRECEIVED_MODEL_VOLATILITY	Attach a volatility to the received cms leg
ADX_PTR_MAP	Map to be attached
ADX_PTR_MODEL_DISCOUNTRATE	Discount rate model attached to the instrument
ADX_PTR_MODEL_FX	Fx Model to be attached
ADX_PTR_MODEL_REFINANCINGRATE	Refinancing rate model attached to the instrument
ADX_PTR_MODEL_REINVESTEDRATE	Reinvested rate model attached to the instrument
ADX_PTR_MODEL_VOLATILITY	Volatility Model attached to the equity

See also

- "IAdxInstrument Interface" on page 170

AdxAttrLeg

The `AdxAttrLeg` enumeration defines the set of attributes used by the components derived from the `IAdxLeg` interface. The attributes that describe the component properties are called “Level #1 Attributes” and prefixed ADX_ATTR1. The calculated attributes that can be retrieved at the object level are called “Level #2 Attributes” and prefixed ADX_ATTR2.

Level #1 Attributes

ADX_ATTR1D_LEG_START	Starting date of the leg
ADX_ATTR1D_LEG_DATED	First accrual date of the leg
ADX_ATTR1D_LEG_FRCD	First regular coupon date of the leg
ADX_ATTR1D_LEG_LRCD	Last regular coupon date of the leg
ADX_ATTR1D_LEG_FAD	First amortization date
ADX_ATTR1E_LEG_MDADJ	Maturity date adjustment
ADX_ATTR1E_LEG_PEX	Notional principal exchange parameter
ADX_ATTR1E_LEG_NC	Attribute to specify whether the capital is normalized or not
ADX_ATTR1E_LEG_IAC	Flag to separate or not interest and capital cash flows
ADX_ATTR1E_LEG_IC	Type of the first coupon period of the leg
ADX_ATTR1E_LEG_EMC	End of month convention

ADX_ATTR1E_LEG_DMC	Date moving convention
ADX_ATTR1E_LEG_CFADJ	Cash flow adjustment method
ADX_ATTR1E_LEG_ALIMIT	Accrued limit type
ADX_ATTR1E_LEG_ACC	Accrued calculation method
ADX_ATTR1E_LEG_CCM	Coupon calculation method
ADX_ATTR1E_LEG_REFDATE	Reference date for coupon date generation (maturity date or issue date)
ADX_ATTR1E_LEG_ARNDMODE	Accrued rounding mode definition
ADX_ATTR1E_LEG_RT	Nominal reimbursement type
ADX_ATTR1E_LEG_FRQ	Coupon frequency
ADX_ATTR1E_LEG_PX	Price type (Clean or Gross)
ADX_ATTR1E_LEG_CALL	Attribute to specify that the leg is callable
ADX_ATTR1E_LEG_PUT	Attribute to specify that the leg is putable
ADX_ATTR1E_LEG_CRNDMODE	Coupon rounding mode definition
ADX_ATTR1F_LEG_PSCF	Notional principal amount exchanged at start date
ADX_ATTR1F_LEG_PMCF	Notional principal amount exchanged at maturity date
ADX_ATTR1F_LEG_INTCAP	Capitalization period
ADX_ATTR1F_LEG_FCV	Value for first odd coupon
ADX_ATTR1F_LEG_RP	Redemption value
ADX_ATTR1F_LEG_CRND	Coupon rounding definition
ADX_ATTR1F_LEG_ARND	Accrued rounding definition
ADX_ATTR1F_LEG_NOTIONAL	Notional Value
ADX_ATTR1F_LEG_AMORT	Capital amortized value
ADX_ATTR1F_LEG_PPMT	Capital partial payment value
ADX_ATTR1L_LEG_PDELAY	Number of days for payment delay
ADX_ATTR1S_LEG_MATURITY_CODE	Maturity code of the leg
ADX_ATTR1S_LEG_XD	Ex-Dividend rule
ADX_ATTR1S_LEG_LOCKOUT	Lockout rule
ADX_ATTR1S_LEG_IDX	Index reference for historical database
ADX_ATTR1S_LEG_PXRND	Price rounding
ADX_ATTR1S_LEG_YLDRND	Yield rounding

Level #2 Attributes

ADX_ATTR2D_LEG_PREVCPNDATE	Leg Previous CouponDate
ADX_ATTR2D_LEG_NEXTCPNDATE	Leg Next Coupon Date
ADX_ATTR2D_LEG_EXDIVIDEND	Leg Ex-dividend Date

ADX_ATTR2F_LEG_ACCRUED	Leg Accrued
ADX_ATTR2F_LEG_NEXTCPN	Leg Next Coupon
ADX_ATTR2L_LEG_ACCRUEDDAYS	Leg Accrued days
ADX_ATTR2L_LEG_CPNNUMBER	Leg Coupon Number
ADX_ATTR2S_LEG_NOTIONALS_CHEDULE	Leg Notional

Identifier

ADX_PTR_MODEL_RISK	Risk model attached to the leg

See also

- "IAdxLeg Interface" on page 205

AdxAttrMC

The AdxAttrMC enumeration defines the set of attributes used by the AdxCalcMethod component. These attributes, which describe the component properties, are called “Level #1 Attributes" and prefixed ADX_ ATTR1.

Level #1 Attributes

ADX_ATTR1E_MC_RANDOM	Type of random generator used for Monte Carlo simulation
ADX_ATTR1E_MC_RANDOM	Type of random generator used for Monte Carlo simulation
ADX_ATTR1L_MC_NB_SIM	Number of simulation used for Monte Carlo pricing
ADX_ATTR1L_MC_NB_TIMESTEPS	Number of timesteps used for Monte Carlo pricing
ADX_ATTR1L_MC_NBEX	Number of exercises for an auto-flexi cap/floor

See also

- "IAdxCalcMethod Interface" on page 172

AdxAttrNToDefaultCDS

The AdxAttrNToDefaultCDS enumeration defines the set of attributes used by the AdxNToDefaultCDS component. These attributes, which describe the component properties, are called “Level #1 Attributes" and prefixed ADX_ATTR1. The attributes that can only be retrieved after computation are called “Level #3 Attributes" and prefixed ADX_ATTR3.

Level #1 Attributes

ADX_ATTR1L_NDEFAULT_NBDEFAULT	Number of default defining the Nth To Default Cds
ADX_ATTR1R_NDEFAULT_NOMINAL_ARRAY	Nominal Array of the Nth To Default Cds

Level #3 Attributes

ADX_ATTR3F_NDEFAULT_NPV	Npv of the Nth To Default

ADX_ATTR3F_NDEFAULT_SPREAD	Nth To Default Cds Spread

See also

"AdfinX Analytics Objects" on page 228

AdxAttrOpBinary

The AdxAttrOpBinary enumeration defines the set of attributes used by the AdxOpBinary component. These attributes, which describe the component properties are called “Level #1 Attributes” and prefixed ADX_ATTR1.

Level #1 Attributes

ADX_ATTR1E_EXO_BINARYTYPE	Type of the Binary option
ADX_ATTR1E_EXO_TOUCH	TOUCH option
ADX_ATTR1F_EXO_CASH_AMOUNT	Cash amount

See also

- "AdxOpBinary" on page 270
- "IAdxOpBinary Interface" on page 208

AdxAttrOpLookBack

The AdxAttrOplookBack enumeration defines the set of attributes used by the AdxOpLookBack component. These attributes, which describe the component properties, are called “Level #1 Attributes” and prefixed ADX_ATTR1.

Level #1 Attributes

ADX_ATTR1E_EXO_LOOKTYPE	Type of the LookBack option
ADX_ATTR1F_MINMAX	Up to date Min or Max value for the spot
ADX_ATTR1R_LADDER	Array of ladder value used in LadderOption

See also

- "AdxOpLookBack" on page 272
- "IAdxOpLookBack Interface" on page 208

AdxAttrOption

The AdxAttrOption enumeration defines the set of attributes used by the AdxOption component. The attributes that describe the component properties are called “Level #1 Attributes” and prefixed ADX_ATTR1. The attributes that can only be retrieved after computation are called “Level #3 Attributes” and prefixed ADX_ATTR3. The underlying identifier is prefixed ADX_PTR.

Level #1 Attributes

ADX_ATTR1D_OPT_DELIVERY	Delivery Date
ADX_ATTR1D_OPT_SPOT	Option Spot
ADX_ATTR1E_OPT_CALL	Option Type Call
ADX_ATTR1E_OPT_CAS	Array Contain Type
ADX_ATTR1E_OPT_DILUTION	Option Dilution
ADX_ATTR1E_OPT_DMC	Date Moving Convention
ADX_ATTR1E_OPT_EXM	Exercise Mode
ADX_ATTR1E_OPT_FIRST_LET	Option Floorlet/Caplet
ADX_ATTR1E_OPT_PUT	Option Type Put
ADX_ATTR1E_OPT_STODCB	Stochastic Day Count Basis for expiry period calculation
ADX_ATTR1E_OPT_UO	Underlying Option Type (for Compound Option)
ADX_ATTR1F_OPT_ALPHA	Alpha Parameter for Power Option
ADX_ATTR1F_OPT_BARRIER_DOWN_PRICE	Price for single barrier or down barrier
ADX_ATTR1F_OPT_BARRIER_UP_PRICE	Price for up barrier
ADX_ATTR1F_OPT_CAP	Cap in Trinomial Tree
ADX_ATTR1F_OPT_CONVRATIO	Conversion Ratio
ADX_ATTR1F_OPT_EXCHANGE_RATE	Rate Exchange
ADX_ATTR1F_OPT_FXLINKED	Quanto Type Option
ADX_ATTR1F_OPT_NBSTOCK	Number of Share Outstanding
ADX_ATTR1F_OPT_NBWARRANT	Number of Warrant Outstanding
ADX_ATTR1F_OPT_POWER	Power Parameter for Power Option
ADX_ATTR1F_OPT_STRIKE	Strike Price
ADX_ATTR1E_OPT_RATES	Risk Free Rate
ADX_ATTR1R_OPT_DATE_STRIKE	Array of Date and Strike
ADX_ATTR1R_OPT_FORWARD_DATE	ForwardDate Array for Cliquet type option
ADX_ATTR1S_OPT_CLDR	Calendar Type

Level #3 Attributes

ADX_ATTR3F_OPT_BREAK_EVEN_TIME	Break Even Time Calculation
ADX_ATTR3F_OPT_CHARM	Charm Calculation
ADX_ATTR3F_OPT_COLOR	Color Calculation
ADX_ATTR3F_OPT_DELTA	Delta Calculation
ADX_ATTR3F_OPT_DELTA_FOREIGN	Delta Foreign Calculation
ADX_ATTR3F_OPT_DELTA_FORWARD	Delta Forward Calculation
ADX_ATTR3F_OPT_DELTA1	Delta1 Calculation
ADX_ATTR3F_OPT_DELTA2	Delta2 Calculation

ADX_ATTR3F_OPT_DUAL_DELTA	Dual Delta Calculation
ADX_ATTR3F_OPT_DUAL_GAMMA	Dual Gamma Calculation
ADX_ATTR3F_OPT_DUAL_THETA	Dual Theta Calculation
ADX_ATTR3F_OPT_DVEGADTIME	DvegaDtime Calculation
ADX_ATTR3F_OPT_GAMMA	Gamma Calculation
ADX_ATTR3F_OPT_GAMMA_FORWARD	Gamma Forward Calculation
ADX_ATTR3F_OPT_GEARING	Gearing Calculation
ADX_ATTR3F_OPT_GAMMA1	Gamma1 Calculation
ADX_ATTR3F_OPT_GAMMA2	Gamma2 Calculation
ADX_ATTR3F_OPT_GEARING	Gearing Calculation
ADX_ATTR3F_OPT_IMPVOL	Implied Volatility Calculation
ADX_ATTR3F_OPT_PREMIUM	Premium Calculation
ADX_ATTR3F_OPT_RHO	Rho Calculation
ADX_ATTR3F_OPT_RHO_FOREIGN	Rho Foreign Calculation
ADX_ATTR3F_OPT_RHO1	Rho1 Calculation
ADX_ATTR3F_OPT_RHO2	Rho2 Calculation
ADX_ATTR3F_OPT_SPEED	Speed Calculation
ADX_ATTR3F_OPT_THETA	Theta Calculation
ADX_ATTR3F_OPT_THETA_FOREIGN	Theta Foreign Calculation
ADX_ATTR3F_OPT_ULTIMA	Ultima Calculation
ADX_ATTR3F_OPT_VANNA	Vanna Calculation
ADX_ATTR3F_OPT_VEGA	Vega Calculation
ADX_ATTR3F_OPT_VEGA_FOREIGN	Vega Foreign Calculation
ADX_ATTR3F_OPT_VEGA1	Vega1 Calculation
ADX_ATTR3F_OPT_VEGA2	Vega2 Calculation
ADX_ATTR3F_OPT_VOLGA	Volga Calculation
ADX_ATTR3F_OPT_ZOMMA	Zomma calculation

Instrument identifier

ADX_PTR_INSTRUMENT_UNDERLYING1	Underlying attached to the option

See also

- "AdxOption" on page 275
- "IAdxOption Interface" on page 209

AdxAttrParse

The `AdxAttrParse` enumeration defines the set of attributes used by the `AdxParse` component. The attributes that describe the component properties are called “Level #1 Attributes” and prefixed ADX_

ATTR1.The attributes that can only be retrieved after computation are called “Level #3 Attributes” and prefixed ADX_ATTR3.

Level #1 Attributes

ADX_ATTR1E_PARSE_PDF	Bid Ask Format
ADX_ATTR1E_PARSE_PDT	Parsing data type
ADX_ATTR1L_PARSE_FD	Fraction denominator for Bond Parsing
ADX_ATTR1L_PARSE_LEN	Number of characters of the substring to parse
ADX_ATTR1L_PARSE_POS	Position of the first character of the substring to parse
ADX_ATTR1S_PARSE_STR	Input String

Level #3 Attribute

ADX_ATTR3R_PARSE_RESULT	Output array

See also

- "AdxParse" on page 278
- "IAdxParse Interface" on page 210

AdxAttrRainbow

The AdxAttrRainbow enumeration defines the set of attributes used by the AdxRainbow component. These attributes, which describe the component properties, are called “Level #1 Attributes” and prefixed ADX_ATTR1.

Level #1 Attributes

ADX_ATTR1F_EXO_DOUBLESTRIKE	Double Strike value
ADX_ATTR1F_EXO_RAINTYPE	Rainbow type

See also

- "AdxRainbow" on page 279
- "IAdxRainbow Interface" on page 210

AdxAttrRateModel

The AdxAttrRateModel enumeration defines the set of attributes used by the AdxRateModel component. These attributes, which describe the component properties, are called “Level #1 Attributes” and prefixed ADX_ATTR1.

Level #1 Attributes

ADX_ATTR1D_RATEMODEL_STARTDATE	Start Date of the rate model

ADX_ATTR1E_RATEMODEL_CLDRADJ	Calendar Adjustment
ADX_ATTR1E_RATEMODEL_DCB	Day Count Basis
ADX_ATTR1E_RATEMODEL_DF	Zero Coupon type {Discount factor or Rates}
ADX_ATTR1E_RATEMODEL_EY	Equivalent yield
ADX_ATTR1E_RATEMODEL_FRQ	Compounding frequency
ADX_ATTR1E_RATEMODEL_IAY	Inflation adjustment flag {NO, YES}
ADX_ATTR1E_RATEMODEL_IM	Interpolation Method for rates LIN/CUBD/CUBR
ADX_ATTR1E_RATEMODEL_IMVOL	Interpolation Method for volatility LIN/CUBD/CUBR
ADX_ATTR1E_RATEMODEL_MODELTYPE	Model Type (Yield To Maturity, Zero-Coupon, Vasicek-Fong, Black, Derman, and Toy, Black and Scholes, and Hull and White}
ADX_ATTR1E_RATEMODEL_ND	Null date processing {DIS: discard, ERR: error}
ADX_ATTR1E_RATEMODEL_OBC	Extrapolation method
ADX_ATTR1E_RATEMODEL_RATETYPE	Rate type {Effective, Discount, Continuous, Money Market}
ADX_ATTR1E_RATEMODEL_TO	Curve type (DF or RATE)
ADX_ATTR1E_RATEMODEL_VOLTYPE	Volatility Type {SR: Short Rate, ZC: Zero Coupon}
ADX_ATTR1F_RATEMODEL_SHIFT	Shift value on the rate curve
ADX_ATTR1L_RATEMODEL_YB	Year basis
ADX_ATTR1R_BGMRATEMODEL_VOLATPARAMS	Volat Params
ADX_ATTR1R_RATEMODEL_ARRAY	Rate model data array
ADX_ATTR1S_RATEMODEL_CLDR	Calendar
ADX_ATTR1E_RATEMODEL_LLP	Last Linear Period
ADX_ATTR1S_RATEMODEL_LNP	Last Nominal Period

See also

- "AdxRateModel" on page 282
- "IAdxRateModel Interface" on page 211

AdxAttrRepo

The `AdxAttrRepo` enumeration defines the set of attributes used by the `AdxRepo` component. The attributes that describe the component properties are called “Level #1 Attributes” and prefixed ADX_ATTR1. The attributes that can only be retrieved after computation are called “Level #3 Attributes” and prefixed ADX_ATTR3. The identifiers are prefixed ADX_PTR.

Level #1 Attributes

`ADX_ATTR1E_REPO_FV_TYPE`	Future value type {C:CLEAN, G:GROSS, Y:YIELD}
`ADX_ATTR1E_REPO_FV_XD`	Ex-dividend calculation for future value {NO, YES}
`ADX_ATTR1E_REPO_ICINCL`	Flag to indicate if intermediate coupons are exchanged during the durations of the repo {NO, YES}
`ADX_ATTR1E_REPO_NPV_TYPE`	Net Present value type {C:CLEAN, G:GROSS, Y:YIELD}
`ADX_ATTR1E_REPO_NPV_XD`	Ex-dividend calculation for net present value {NO, YES}
`ADX_ATTR1F_REPO_FV_RP`	Future value redemption ratio {i with i as numeric (1=100%)}
`ADX_ATTR1F_REPO_NPV_RP`	Present value redemption ratio {i with i as numeric (1=100%)}

Level #3 Attributes

`ADX_ATTR3F_REPO_FV`	Future value
`ADX_ATTR3F_REPO_NPV`	Present value
`ADX_ATTR3F_REPO_REPO_RATE`	Repo Rate

Identifiers

`ADX_PTR_INSTRUMENT_COLLATERAL`	Collateral attached to the repo
`ADX_PTR_MODEL_REPO_RATE`	Discount rate model attached to the Repo

See also

- "AdxRepo" on page 282
- "IAdxRepo Interface" on page 211

AdxAttrRiskModel

The `AdxAttrRiskModel` enumeration defines the set of attributes used by the `AdxRiskModel` component. These attributes, which describe the component properties, are called “Level #1 Attributes” and prefixed ADX_ATTR1.

Level #1 Attributes

`ADX_ATTR1D_RISKMODEL_STARTDATE`	Model Start Date
`ADX_ATTR1E_RISKMODEL_COPULA_TYPE`	Copula Type
`ATTR1E_RISKMODEL_CREDITEVENT`	Credit event type
`ATTR1E_RISKMODEL_IM`	Interpolation method for probability curves
`ATTR1E_RISKMODEL_MODELTYPE`	Model type (Cox, Ingersoll, and Ross model or probability curve)
`ATTR1E_RISKMODEL_ND`	Null date processing
`ATTR1E_RISKMODEL_OBC`	Extrapolation method
`ADX_ATTR1E_RISKMODEL_PROBA_INTERP`	Method for the probability curve
`ATTR1E_RISKMODEL_RATING`	Rating
`ADX_ATTR1F_RISKMODEL_COPULA_PARAMETER`	Copula parameter like freedom degrees
`ATTR1F_RISKMODEL_RECOVERY`	Recovery rate
`ADX_ATTR1L_RISKMODEL_COPULA_FACTOR`	Number of factors in the copula
`ATTR1L_RISKMODEL_NBDAYS`	Number of days for American credit default swap pricing
`ATTR1R_RISKMODEL_ARRAY`	Risk model data array
`ADX_ATTR1R_RISKMODEL_COPULA_CORRELATION`	Copula Correlations
`ADX_ATTR1R_RISKMODEL_COPULA_PARAMETER`	Copula parameter like freedom degrees
`ADX_ATTR1R_RISKMODEL_INST_ARRAY`	Instrument Array
`ADX_ATTR1R_RISKMODEL_MULTI_ARRAY`	Copula Risk Model Data Array of all the names
`ADX_ATTR1R_RISKMODEL_MULTI_RECOVERY`	Recovery rates of all names
`ADX_ATTR1R_RISKMODEL_PROBA`	Return only probability

See also

- "AdxRiskModel" on page 285
- "IAdxRiskModel Interface" on page 213

AdxAttrSchedule

The `AdxAttrSchedule` enumeration defines the set of attributes used by the `AdxSchedule` component. These attributes, which describe the component properties, are called “Level #1 Attributes" and prefixed ADX_ATTR1. The attributes that can only be retrieved after computation are called “Level #3 Attributes" and prefixed ADX_ATTR3. The model identifiers are prefixed ADX_PTR.

ADX_ATTR1D_SCHEDULE_CALCDATE	Calculation date
ADX_ATTR1E_FLOWEXT_CALC_CLDRADJ	Calendar adjustment
ADX_ATTR1E_FLOWEXT_CALC_DMC	Date moving convention
ADX_ATTR1E_FLOWEXT_CALC_EMC	End-of-month convention
ADX_ATTR1E_FLOWEXT_CONVBIAS	Convexity bias
ADX_ATTR1E_FLOWEXT_FIX_CLDRADJ	Calendar adjustment
ADX_ATTR1E_FLOWEXT_FIX_DMC	Date moving convention
ADX_ATTR1E_FLOWEXT_FIX_EMC	End-of-month convention
ADX_ATTR1E_FLOWEXT_FVINPUT	
ADX_ATTR1E_FLOWEXT_IEDINPUT	
ADX_ATTR1E_FLOWEXT_ISDINPUT	
ADX_ATTR1E_FLOWEXT_PAY_CLDRADJ	Calendar adjustment
ADX_ATTR1E_FLOWEXT_PAY_DMC	Date moving convention
ADX_ATTR1E_FLOWEXT_PAY_EMC	End-of-month convention
ADX_ATTR1E_FLOWEXT_PROBAINPUT	
ADX_ATTR1E_FLOWEXT_RRINPUT	
ADX_ATTR1F_SCHEDULE_ACCRUED	Calculating Accrued
ADX_ATTR1F_SCHEDULE_CALCULATE_ALL	Calculate all
ADX_ATTR1F_SCHEDULE_CONVEXITY	Calculating Convexity
ADX_ATTR1F_SCHEDULE_CONVEXITY_ARRAY	Calculating Convexity array, one column for each currency
ADX_ATTR1F_SCHEDULE_CURRENCY2_CURRENCY1_SPOT	Fx Spot Currency2/Currency 1
ADX_ATTR1F_SCHEDULE_CURRENCY3_CURRENCY1_SPOT	Fx Spot Currency3/Currency 1
ADX_ATTR1F_SCHEDULE_DURATION	Duration
ADX_ATTR1F_SCHEDULE_DURATION_ARRAY	Calculating Duration
ADX_ATTR1F_SCHEDULE_NPV	Calculating Npv
ADX_ATTR1F_SCHEDULE_PVBP	Calculating PVBP
ADX_ATTR1F_SCHEDULE_PVBP_ARRAY	Calculating PVBP array
ADX_ATTR1F_SCHEDULE_PVBPCRV	PVBP Curve
ADX_ATTR1F_SCHEDULE_TARGET_NPV	Input Target to implicit the yield
ADX_ATTR1F_SCHEDULE_YIELD	Yield to implicit
ADX_ATTR1L_SCHEDULE_SELECT_GROUP_NUMBER	Selection group number
ADX_ATTR1S_FLOWEXT_CALC_CLDR	
ADX_ATTR1S_FLOWEXT_FIX_CLDR	

ADX_ATTR1S_FLOWEXT_PAY_CLDR	
ADX_ATTR1S_SCHEDULE_CALCSTRUCT_RESULT	
ADX_ATTR1S_SCHEDULE_CALCSTRUCT_YIELDCCY	
ADX_ATTR1S_SCHEDULE_CALCSTRUCTURE	Calculation structure
ADX_ATTR1S_SCHEDULE_CURRENCY1	First currency (base currency)
ADX_ATTR1S_SCHEDULE_CURRENCY1_FLOATING_RATESTRUCTURE	Floating rate structure for currency 1
ADX_ATTR1S_SCHEDULE_CURRENCY1_RATESTRUCTURE	Rate structure for currency 1
ADX_ATTR1S_SCHEDULE_CURRENCY2	2nd currency
ADX_ATTR1S_SCHEDULE_CURRENCY2_FLOATING_RATESTRUCTURE	Floating rate structure for currency 2
ADX_ATTR1S_SCHEDULE_CURRENCY2_RATESTRUCTURE	Rate structure for currency 2
ADX_ATTR1S_SCHEDULE_CURRENCY3	3rd currency (base currency)
ADX_ATTR1S_SCHEDULE_CURRENCY3_FLOATING_RATESTRUCTURE	Floating rate structure for currency 3
ADX_ATTR1S_SCHEDULE_CURRENCY3_RATESTRUCTURE	Rate structure for currency 3
ADX_ATTR1S_SCHEDULE_DERIV_RESULT	Result of AdScheduleDeriv Function: Pvbp, Convexity , Duration,NPV or ALL
ADX_ATTR1S_SCHEDULE_NPV_CURRENCY	Currency to use for countervaluation of global Npv (default is Curency 1)
ADX_ATTR1S_SCHEDULE_SELECT_CURRENCY	Selection currency
ADX_ATTR1S_SCHEDULE_YIELD_CURRENCY	Yield currency

Level #3 Attributes

ADX_ATTR3R_SCHEDULE_CURRENCY1_FLOATING_RATE_CURVE	Floating rate curve for currency 1
ADX_ATTR3R_SCHEDULE_CURRENCY1_INFLATION_ARRAY	Inflation Array for currency 1
ADX_ATTR3R_SCHEDULE_CURRENCY1_RATEVOLATILITY	Rate volatility Array for currency 1
ADX_ATTR3R_SCHEDULE_CURRENCY1_YIELD_CURVE	Yield curve for currency 1
ADX_ATTR3R_SCHEDULE_CURRENCY2_CURRENCY1_CORRELATION	Correlation between currency2 and Currency1
ADX_ATTR3R_SCHEDULE_CURRENCY2_CURRENCY1_FX_VOLATILITY	Fx volatility for Currency2/Currency 1
ADX_ATTR3R_SCHEDULE_CURRENCY2_FLOATING_RATE_CURVE	Floating rate curve for currency 2
ADX_ATTR3R_SCHEDULE_CURRENCY2_INFLATION_ARRAY	Inflation Array for currency 2

ADX_ATTR3R_SCHEDULE_CURRENCY2_RATEVOLATILITY	Rate volatility Array for currency 2
ADX_ATTR3R_SCHEDULE_CURRENCY2_YIELD_CURVE	Yield curve for currency 2
ADX_ATTR3R_SCHEDULE_CURRENCY3_CURRENCY1_CORRELATION	Correlation between currency 3 and Currency 1
ADX_ATTR3R_SCHEDULE_CURRENCY3_CURRENCY1_FX_VOLATILITY	Fx volatility for Currency 3/Currency 1
ADX_ATTR3R_SCHEDULE_CURRENCY3_FLOATING_RATE_CURVE	Floating rate curve for currency 3
ADX_ATTR3R_SCHEDULE_CURRENCY3_INFLATION_ARRAY	Inflation Array for currency 3
ADX_ATTR3R_SCHEDULE_CURRENCY3_RATEVOLATILITY	Rate volatility Array for currency 3
ADX_ATTR3R_SCHEDULE_CURRENCY3_YIELD_CURVE	Yield curve for currency 3
ADX_ATTR3R_SCHEDULE_LIBOR_MAP_ARRAY	Libor Map
ADX_ATTR3R_SCHEDULE_LIBOR_VALUES_ARRAY	Libor values array
ADX_ATTR3R_SCHEDULE_SCHEDULE_ARRAY	Calculating cashflows
ADX_ATTR3R_SCHEDULE_SCHEDULE_ARRAY_WITH_GREEKS	Calculating cashflows with greeks

Identifier

ADX_PTR_MODEL_LIBORMAP	Index map to be attached

See also

- "AdfinX Analytics Objects" on page 228

AdxAttrSwap

The `AdxAttrSwap` enumeration defines the set of attributes used by the `AdxSwap` component. The attributes that describe the component properties are called “Level #1 Attributes” and are prefixed ADX_ATTR1. The calculated attribute that can be retrieved at the object level is called “Level #2 Attribute” and prefixed ADX_ATTR2. The attributes that can only be retrieved after computation are called “Level #3 Attributes” and prefixed ADX_ATTR3. Model and Instrument identifiers are prefixed ADX_PTR.

Level #1 Attributes

ADX_ATTR1E_SWAP_ENABLEDIFFERENTLEGMATURITY	Enable different leg maturity for swaps
ADX_ATTR1E_SWAP_LBOTH	Swap attribute specification flag
ADX_ATTR1E_SWAP_LFIXED	Fixed leg attribute flag
ADX_ATTR1E_SWAP_LFLOAT	Float leg attribute flag
ADX_ATTR1E_SWAP_PAID	Float leg attribute flag
ADX_ATTR1E_SWAP_LRECEIVED	Received leg attribute flag
ADX_ATTR1E_SWAP_LTYPE	Type of the current leg {FIXED, FLOAT}
ADX_ATTR1E_SWAP_LPAID	Paid leg attribute flag

ADX_ATTR1E_SWAP_PAID	Paid leg attribute flag
ADX_ATTR1E_SWAP_PAID_FRQ	Coupon frequency for the paid leg
ADX_ATTR1E_SWAP_PEX	Notional principal exchange parameter
ADX_ATTR1E_SWAP_PXT	Type of Npv specified for the swap. (Received, Paid or Swap Npv)
ADX_ATTR1E_SWAP_RECEIVED_FRQ	Coupon frequency for the received leg
ADX_ATTR1E_SWAP_TYPE	Type of credit derivative swaps
ADX_ATTR1F_PAID_CURRENT_FLOAT_INDEX	Current index value of the paid leg of the swap
ADX_ATTR1F_PAID_LEG_RP	Redemption price of the paid leg of the swap
ADX_ATTR1F_RECEIVED_CURRENT_FLOAT_INDEX	Current index value of the received leg of the swap
ADX_ATTR1F_RECEIVED_LEG_RP	Redemption price of the received leg of the swap
ADX_ATTR1F_SWAP_PMSPOT	Notional principal exchange rate at the swap maturity date
ADX_ATTR1F_SWAP_PSSPOT	Notional principal exchange rate at the swap start date
ADX_ATTR1S_PAID_LEG_CONVBIAS	Convexity bias for paid leg
ADX_ATTR1S_PAID_LEG_ISQUANTO	Paid Leg is a Quanto (floating reference currency is different than swap currency)
ADX_ATTR1S_RECEIVED_LEG_CONVBIAS	Convexity bias for received leg
ADX_ATTR1S_RECEIVED_LEG_ISQUANTO	Paid Leg is a Quanto (floating reference currency is different than swap currency)
ADX_ATTR1S_SWAP_CROSS	Cross-currency parameter for currency swaps
ADX_ATTR1S_SWAP_FIX_RATE	Fixed or spread rate if not passed directly by the function
ADX_ATTR1S_SWAP_PAID_CURRENCY	Currency associated to paid leg of the swap Swap
ADX_ATTR1S_SWAP_QUANTO_CURRENCY	Currency associated to Quanto Swap (payment made in that currency for both legs)
ADX_ATTR1S_SWAP_RECEIVED_CURRENCY	Currency associated to received leg of the Swap

Level #2 Attributes

ADX_ATTR2R_CDS_FIXED_CASHFLOWS	Cash Flow value for the Cds Spread Leg
ADX_ATTR2R_SWAP_ACCRUED	Accrued leg for the swap, the paid leg and the received leg
ADX_ATTR2R_SWAP_CASHFLOWS	Cash flows of the swap
ADX_ATTR2R_SWAP_FIXED_DATES	Coupon dates of the fixed leg
ADX_ATTR2R_SWAP_FLOAT_DATES	Coupon dates of the float leg
ADX_ATTR2R_SWAP_PAID_DATES	Coupon dates of the paid leg
ADX_ATTR2R_SWAP_RECEIVED_DATES	Coupon dates of the received leg

ADX_ATTR3F_SWAP_DURATION	The duration of the swap
ADX_ATTR3F_SWAP_FIXED_NPV	The net present value of fixed leg
ADX_ATTR3F_SWAP_FIXED_RATE	Fixed Interest Rate
ADX_ATTR3F_FLOAT_NPV	The net present value of the float leg
ADX_ATTR3F_SWAP_FLOAT_SPREAD	The net present value of the float leg
ADX_ATTR3F_SWAP_INFLATION_SPREAD	Calculate the spread over the inflation rate
ADX_ATTR3F_SWAP_LEVEL	Swap Level or Swap Numeraire
ADX_ATTR3F_SWAP_NPV	The net present value of the swap
ADX_ATTR3F_SWAP_OPPREMIUM	Callable/Puttable Swap Option Premium
ADX_ATTR3F_SWAP_PAID_DUR	The duration of the paid leg
ADX_ATTR3F_SWAP_PAID_FIXED_RATE	The fixed rate that is paid
ADX_ATTR3F_SWAP_PAID_NPV	The net present value of the paid leg
ADX_ATTR3F_SWAP_PAID_RATE	Spread or fixed rate that is paid
ADX_ATTR3F_SWAP_PAID_SPREAD	The spread that is paid
ADX_ATTR3F_SWAP_PARRATE	Callable/Puttable Swap Par Rate
ADX_ATTR3F_SWAP_PAID_PRICE	The spread or fixed rate that is paid
ADX_ATTR3F_SWAP_RANN	Risky Annuity
ADX_ATTR3F_SWAP_RECEIVED_DUR	The duration of the received leg
ADX_ATTR3F_SWAP_RECEIVED_FIXED_RATE	The fixed rate paid that is received
ADX_ATTR3F_SWAP_RECEIVED_NPV	The net present value of the received leg
ADX_ATTR3F_SWAP_RECEIVED_RATE	Spread or fixed rate that is received
ADX_ATTR3F_SWAP_RECEIVED_SPREAD	The spread that is received
ADX_ATTR3F_SWAP_RHO	Rho of recovery rate
ADX_ATTR3F_SWAP_THETA	Theta
ADX_ATTR3R_SWAP_CONVEXITY	Calculates the convexity of the swap and of each of its legs
ADX_ATTR3F_SWAP_RECEIVED_PRICE	The spread or fixed rate that is received
ADX_ATTR3R_SWAP_DURATION	Calculates the fixed/float leg and global swap duration
ADX_ATTR3R_SWAP_PVBP	Calculates price variation per basis point variation on the risk-free rate curve */
ADX_ATTR3R_SWAP_RBPV	CDS rate BPV
ADX_ATTR3R_SWAP_RDUR	Risky Duration
ADX_ATTR3R_SWAPOINT_EXTEND	Extend Swap point curve for currency swap

Model Identifiers

ADX_PTR_MODEL_PAID_DISCOUNTRATE	Discount Rate for the paid leg
ADX_PTR_MODEL_RECEIVED_DISCOUNTRATE	Discount Rate for the received leg

Instrument Identifiers

ADX_PTR_INSTRUMENT_PAID	Instrument to be attached as the paid leg
ADX_PTR_INSTRUMENT_RECEIVED	Instrument to be attached as the received leg

See also

- "AdxSwap" on page 289
- "IAdxSwap Interface" on page 220

AdxAttrTermStructure

The AdxAttrTermStructure enumeration defines the set of attributes used by the AdxTermStructure component. The attributes that describe the component properties are called “Level #1 Attributes” and prefixed ADX_ATTR1. The calculated attribute that can be retrieved at the object level is called "Level #2 Attribute" and prefixed ADX_ATTR2.

Level #1 Attributes

ADX_ATTR1D_BGM_CALCDATE	CalcDate
ADX_ATTR1D_TERMSTRUCTURE_YCSTARTDATE	TermStructure start date
ADX_ATTR1E_BGM_CALIBRATION_TYPE	Calibration Type
ADX_ATTR1E_BGM_SEARCH_TYPE	Search Type
ADX_ATTR1E_BSPLINE_MODEL	Model
ADX_ATTR1E_BSPLINE_VOL	Volatility model
ADX_ATTR1E_TERMSTRUCTURE_APPROX	Approximation Type for Cox, Ingersoll and Ross calibration
ADX_ATTR1E_TERMSTRUCTURE_INSTTYPE	Instrument array type for risk model calibration
ADX_ATTR1E_TERMSTRUCTURE_MDWA	Volatility model for B Spline methods
ADX_ATTR1E_TERMSTRUCTURE_MODELTYPE	Model Type (ZC,VF)
ADX_ATTR1E_TERMSTRUCTURE_SKIPINSTRUMENT	Skip instruments between 2 step dates
ADX_ATTR1E_TERMSTRUCTURE_SMOOTH	B Spline method type
ADX_ATTR1E_TERMSTRUCTURE_VFVOL	Volatility value for Vasicek Fong model

ADX_ATTR1E_TERMSTRUCTURE_VOLTYPE	Effective, Discount factor, Continuous, and Money Market
ADX_ATTR1F_BGM_LIBORSPOT	Libor spot
ADX_ATTR1F_TERMSTRUCTURE_BOEAMORT	Amort parameter for Bank of England
ADX_ATTR1F_TERMSTRUCTURE_BOEINF	Time infinite parameter for Bank of England
ADX_ATTR1F_TERMSTRUCTURE_BOEZERO	Time zero parameter for Bank of England
ADX_ATTR1F_TERMSTRUCTURE_HWVOLATILITY	Hull-White volatility for computation of convexity adjustment
ADX_ATTR1F_TERMSTRUCTURE_MEANREVERSION	Mean reversion for computation of convexity adjustment
ADX_ATTR1F_TERMSTRUCTURE_STARTONRATE	Start On Value
ADX_ATTR1E_TERMSTRUCTURE_VFALPHA	Alpha value for Vasicek Fong model
ADX_ATTR1L_BGM_FRQ	Frequency of fixings
ADX_ATTR1L_BGM_NBFACTORS	Number of factors in BGM Calibration
ADX_ATTR1L_BGM_NBHALTONPOINT	Number of Halton Point to consider
ADX_ATTR1L_BGM_NBLOCALMIN	Number of local points to consider
ADX_ATTR1L_TERMSTRUCTURE_NBDAYSTHRESHOLD	Number of days between an step date and an end date of OIS before we move the ois date
ADX_ATTR1L_TERMSTRUCTURE_NBKNOT	Number of points
ADX_ATTR1R_BGM_CAPARRAY	Cap volatilities array in BGM Calibration
ADX_ATTR1R_BGM_RECALIBRATIONLAMBDA	Ci
ADX_ATTR1R_BGM_RELEVANTINDICES	0 = fixed 1 = free
ADX_ATTR1R_BGM_RELEVANTSWAPTIONARRAY	Array which defines which swaption is relevant
ADX_ATTR1R_BGM_SWAPTIONARRAY	Swaption volatilities array in BGM Calibration
ADX_ATTR1R_BGM_SWAPTIONVOLATILITYCALCULATED	Calculated Swaption Volatility Matrix
ADX_ATTR1R_BGM_SWAPTIONVOLATILITYERROR	Swaption Volatility Error Matrix
ADX_ATTR1R_BGM_ZCARRAY	ZC array in BGM Calibration
ADX_ATTR1R_TERMSTRUCTURE_INSTARRAY	Instrument array termstructure
ADX_ATTR1R_TERMSTRUCTURE_STEPDATES	Step Dates

ADX_ATTR1R_TERMSTRUCTURE_VOLARRAY	VolArray in calibration
ADX_ATTR1R_TERMSTRUCTURE_ZCARRAY	ZcArray in calibration
ADX_ATTR1S_TERMSTRUCTURE_CALCMETHOD	CalcMethod used with Trees

Level #2 Attributes

ADX_ATTR2F_BGM_RMSERR	RMS Error
ADX_ATTR2F_BGM_RMSERRMSF	RMS Error for MSF
ADX_ATTR2R_BGM_BGMSTARTPARAMS	Startparams
ADX_ATTR2R_BGM_CAPVOLERROR	Error on cap repricing
ADX_ATTR2R_SWAP_IRS_TO_ZC	Specifies that you convert a swap rate curve to a ZC one

See also

- "IAdxTermStructure Interface" on page 221
- "AdxTermStructure" on page 292

AdxAttrTimeSeries

The AdxAttrTimeSeries enumeration defines the set of attributes used by the AdxTimeSeries component. The attributes that describe the component properties are called “Level #1 Attributes" and prefixed ADX_ATTR1. The calculated attribute that can be retrieved at the object level is called "Level #2 Attribute" and prefixed ADX_ATTR2.

Level #1 Attributes

ADX_ATTR1E_TS_INTERCEPT	Intercept or not
ADX_ATTR1E_TS_INTERCEPTFORSTAT	No Intercept in the Regression but Intercept for Statistics
ADX_ATTR1E_TS_RATEOFRETURN	Translate Price into rate of return
ADX_ATTR1E_TS_SORTEDDATA	Input data sorted in ascending or descending order.
ADX_ATTR1F_TS_LAG	Lag value
ADX_ATTR1R_TS_EXPLANATORY_X	Explanatory(ies) variable(s): X_1, X_2, X_k
ADX_ATTR1R_TS_EXPLICATED_Y	Explicated Y Variable

Level #2 Attributes

ADX_ATTR2F_TS_ADJRSQUARED	Only Adjusted R-Squared value
ADX_ATTR2F_TS_AIC	Only Akaike Information Criterion (AIC) value
ADX_ATTR2F_TS_BIC	Only Bayesian Information Criterion (BIC) value
ADX_ATTR2F_TS_CORRELATION	Only correlation value

ADX_ATTR2F_TS_COVARIANCE	Only covariance value
ADX_ATTR2F_TS_DATA	Only Number of Data value
ADX_ATTR2F_TS_DW	Only Durbin-Watson value
ADX_ATTR2F_TS_FPROBA	Only F-Probability value
ADX_ATTR2F_TS_FSTAT	Only F-Statistic value(s)
ADX_ATTR2F_TS_ML	Only Maximum Likelihood (ML) value*
ADX_ATTR2F_TS_RSQUARED	Only R-Squared value
ADX_ATTR2F_TS_SER	Only Standard Error of Regression value
ADX_ATTR2F_TS_SSE	Only Sum of Squared Errors value
ADX_ATTR2R_TS_COEFFICIENT	Only coefficient(s) value(s)
ADX_ATTR2R_TS_REGRESSION	Global results with all estimated coefficient(s)
ADX_ATTR2R_TS_SECOEF	Only Standard Error of Coefficients value(s)
ADX_ATTR2R_TS_TPROBA	Only Probability(ies) value(s)
ADX_ATTR2R_TS_TSTAT	Only t-statistic value(s)

See also

- "AdxTimeSeries" on page 294

AdxAttrTree

The `AdxAttrTree` enumeration defines the set of attributes used by the `AdxCalcMethod` component. These attributes, which describe the component properties, are called “Level #1 Attributes” and prefixed ADX_ATTR1.

Level #1 Attributes

ADX_ATTR1D_TREE_END	End date of the tree
ADX_ATTR1D_TREE_START	Start date of the tree
ADX_ATTR1E_TREE_ANP	Avoids negative probability in the tree
ADX_ATTR1E_TREE_NBBRANCH	Tree Number of Branches
ADX_ATTR1E_TREE_NBFACTOR	Tree Number of Factors
ADX_ATTR1E_TREE_STEP_ADJUSTMENT	Step Adjustment method flag {NO, YES}
ADX_ATTR1E_TREE_STEP_AVERAGE	Step end date reference {B, BOND, O, OPTION}
ADX_ATTR1E_TREE_STEP_REFERENCE	Step reference
ADX_ATTR1F_TREE_COR	Correlation in Two factors Tree
ADX_ATTR1F_TREE_TIME_STEP	Time frequency between two steps
ADX_ATTR1L_TREE_TITER	Tree Number of Step

See also

- "IAdxCalcMethod Interface" on page 172

AdxAttrVolatilityModel

The `AdxAttrVolatilityModel` enumeration defines the set of attributes used by the `AdxVolatilityModel` component. This attribute, which describes the component properties, is called "Level #1 Attribute" and prefixed ADX_ATTR1. The attributes that can only be retrieved after computation are called "Level #3 Attributes" and prefixed ADX_ATTR3.

Level #1 Attribute

`ADX_ATTR1E_INPUT_PREMIUM`	Premium convention in input
`ADX_ATTR1E_OUTPUT_DELTA`	Delta convention in output
`ADX_ATTR1E_OUTPUT_PREMIUM`	Premium convention in output
`ADX_ATTR1E_VOLATILITY_IM`	Volatility Interpolation Method
`ADX_ATTR1R_SKEWNESS_KURTOSIS`	Skewness Kurtosis
`ADX_ATTR1R_VOLATILITY`	Array of Dates and Volatilities

Level #3 Attribute

`ADX_ATTR3R_SKEWNESS_KURTOSIS_ASK`	Skewness Kurtosis ask
`ADX_ATTR3R_SKEWNESS_KURTOSIS_BID`	Skewness Kurtosis bid
`ADX_ATTR3R_VOLATILITY_ASK`	Ask Volatility surface
`ADX_ATTR3R_VOLATILITY_BID`	Bid Volatility surface

See also

- "AdxVolatilityModel" on page 294
- "IAdxVolatilityModel Interface" on page 223

AdxAccruedCalculation

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxAccruedCalculation")]
enum AdxAccruedCalculation
 {
 ADX_ACC_00              =24,
 ADX_ACC_05g             =31,
 ADX_ACC_0AG             =30,
 ADX_ACC_A0              =22,
 ADX_ACC_A0D             =10,
 ADX_ACC_A0NL            =25,
 ADX_ACC_A4              =37,
 ADX_ACC_A5              =21,
 ADX_ACC_A5D             =8,
```

```
ADX_ACC_A5NL            =13,
ADX_ACC_A5P             =28,
ADX_ACC_AA              =20,
ADX_ACC_AA5             =9,
ADX_ACC_AIX             =39,
ADX_ACC_BB00            = 4,
ADX_ACC_BB0M            = 1,
ADX_ACC_BBA0            = 3,
ADX_ACC_BBA4            = 27,
ADX_ACC_BBA5            = 2,
ADX_ACC_BBA5JD          =43,
ADX_ACC_BBAA            =1,
ADX_ACC_BBAI            =33,
ADX_ACC_BBAX            =32,
ADX_ACC_BBE0            =5,
ADX_ACC_BBEM            =7,
ADX_ACC_BBITL           =19,
ADX_ACC_BBW252          =38,
ADX_ACC_CST             =34,
ADX_ACC_DISC            =40,
ADX_ACC_E0              =23,
ADX_ACC_FRF             =18,
ADX_ACC_GO              =29,
ADX_ACC_IT              =26,
ADX_ACC_IT2             =36,
ADX_ACC_JAP             =11,
ADX_ACC_JAPDEB          =35,
ADX_ACC_JAPMUN          =41,
ADX_ACC_JAPNTT          =42,
ADX_ACC_MM00            =16,
ADX_ACC_MMA0            =14,
ADX_ACC_MMA5            =15,
ADX_ACC_MMAA            =17,
ADX_ACC_MME0            =26,
ADX_ACC_W252            =12
} AdxAccruedCalculation;
```

The `AdxAccruedCalculation` enumeration defines the set of values which determine the accrued interest calculation method.

Values

ADX_ACC_00	For 30/360
ADX_ACC_05G	For 30/365 German
ADX_ACC_0AG	For 30/Actual German
ADX_ACC_A0	For Actual/360
ADX_ACC_A0D	For Actual/360 Day Based
ADX_ACC_A0NL	For (Actual - leap day)/360
ADX_ACC_A4	For Kenyan accrued Actual/364
ADX_ACC_A5	For Actual/365
ADX_ACC_A5D	For Actual/365 Day Based
ADX_ACC_A5NL	For (Actual - leap day)/365
ADX_ACC_A5P	For Actual/365 proportionate
ADX_ACC_AA	For Actual/Actual
ADX_ACC_AA5	For Actual/365.25
ADX_ACC_AIZ	For Russian accrued (Act +1)/Act
ADX_ACC_BB00	For Bond 30/360
ADX_ACC_BB0M	For Bond Basis 30/360 (US) Modified
ADX_ACC_BBA0	For Bond Actual/360
ADX_ACC_BBA4	For Kenyan Bond Actual/364
ADX_ACC_BBA5	For Bond Actual/365
ADX_ACC_BBA5JD	For Bond Actual /365 (6 months) with broken period computed from Date D date and 365 Japan basis
ADX_ACC_BBAA	For Bond Actual/Actual
ADX_ACC_BBAI	For Bond Actual/Actual with broken period computed from issue date
ADX_ACC_BBAX	For Bond Actual/Actual with specific broken adjusted period computed (Eurex compatible)
ADX_ACC_BBE0	For Bond 30E/360 ISMA
ADX_ACC_BBEM	For Bond Basis 30E/360 AIBD Modified
ADX_ACC_BBITL	For Bond Italian
ADX_ACC_BBW252	For Brazilian bond basis W252
ADX_ACC_CST	For Constant: no period calculation taken into account
ADX_ACC_DISC	For Chinese zero coupon bonds
ADX_ACC_E0	For 30E/360 ISMA
ADX_ACC_FRF	For French TMP, T4M, TAM
ADX_ACC_GO	For 30/360 German
ADX_ACC_IT	For Italian (from last coupon date to settlement date, using E0 plus one day)

ADX_ACC_IT2	For Italian modified (from last coupon date to settlement date plus one day, using E0
ADX_ACC_JAP	For Japanese (A5 or A5 plus one day for first coupon)
ADX_ACC_JAPDEB	For non-governmental Japanese bonds (bank debentures)
ADX_ACC_JAPMUN	For non-governmental Japanese bonds (municipal and corporate bonds)
ADX_ACC_JAPNTT	For non-governmental Japanese bonds (NTT bonds)
ADX_ACC_MM00	For Money Market 30/360
ADX_ACC_MMA0	For Money Market Actual/360 - Number of days
ADX_ACC_MMA5	For Money Market Actual/365 - Number of days
ADX_ACC_MMAA	For Money Market Actual/Actual
ADX_ACC_MME0	For Money Market 30E/360 ISMA
ADX_ACC_W252	For 252 working days

See also

- "AdxBond" on page 235
- "AdxConvBond" on page 247
- "AdxFuture" on page 263
- "AdxIIb" on page 266
- "AdxRepo" on page 282
- "AdxSwap" on page 289
- "AdxTermStructure" on page 292

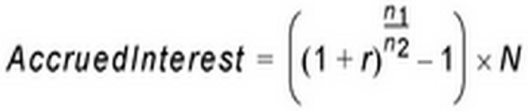
AdxAccruedLimit

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxAccruedLimit")]
enum AdxAccruedLimit
{
ADX_ALIMIT_COUPON                                    =1,
ADX_ALIMIT_NEXT                                      =2,
ADX_ALIMIT_NO                                        =3
} AdxAccruedLimit;
```

The `AdxAccruedLimit` enumeration defines the set of values which determine the accrued interest.

Values

ADX_ALIMIT_COUPON	The resulting accrued interest is the coupon value

ADX_ALIMIT_NEXT	The resulting accrued interest is:  where: n_1 number of accrued days n_2 coupon period (in days) N notional r coupon value
ADX_ALIMIT_NO	The resulting accrued interest is the one calculated

See also

- "AdxBond" on page 235
- "AdxConvBond" on page 247
- "AdxFuture" on page 263
- "AdxIIb" on page 266
- "AdxRepo" on page 282
- "AdxSwap" on page 289
- "AdxTermStructure" on page 292

AdxAccType

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxAccType")]
enum AdxAccType
{
ADX_ACC_DATED                                    =1,
ADX_ACC_NOTYPE                                   =2
} AdxAccType;
```

The AdxAccType enumeration specifies whether a date format has been specified for the accrual start date.

Values

ADX_ACC_DATED	Date format is specified for the accrual start date
ADX_ACC_NOTYPE	No date format is specified for the accrual start date

See also

- "AdxBond" on page 235
- "AdxConvBond" on page 247
- "AdxFuture" on page 263
- "AdxIIb" on page 266
- "AdxRepo" on page 282
- "AdxSwap" on page 289

- "AdxTermStructure" on page 292

AdxAlgorithmInterp

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxAlgorithmInterp")]
enum AdxAlgorithmInterp
{
ADX_ALGO_IM_AFFINE                                    =11,
ADX_ALGO_IM_CUBD                                      =1,
ADX_ALGO_IM_CUBF                                      =14,
ADX_ALGO_IM_CUBR                                      =2,
ADX_ALGO_IM_CUBS                                      =7,
ADX_ALGO_IM_CUBX                                      =8,
ADX_ALGO_IM_EXPO                                      =15,
ADX_ALGO_IM_LIN                                       =3,
ADX_ALGO_IM_LIX                                       =5,
ADX_ALGO_IM_LOG                                       =6,
ADX_ALGO_IM_STEP                                      =9,
ADX_ALGO_IM_STEPX                                     =10,
ADX_ALGO_IM_TSPLINE                                   =12,
ADX_ALGO_IM_TSPLINEX                                  =13,
ADX_ALGO_IM_VOL                                       =4
} AdxAlgorithmInterp;
```

The `AdxAlgorithmInterp` enumeration defines the interpolation mode.

Values

`ADX_ALGO_IM_AFFINE`	Affine interpolation
`ADX_ALGO_IM_CUBD`	Cubic interpolation on the discount factor (extrapolation at the beginning of the curve)
`ADX_ALGO_IM_CUBF`	Cubic differential with extrapolation (constant forward rate)
`ADX_ALGO_IM_CUBR`	Cubic interpolation on rate (extend as flat at the beginning of the curve)
`ADX_ALGO_IM_CUBS`	Cubic Spline interpolation without extrapolation
`ADX_ALGO_IM_CUBX`	Cubic Spline interpolation with extrapolation for out of bound values
`ADX_ALGO_IM_EXPO`	Exponential interpolation (compounded rate)
`ADX_ALGO_IM_LIN`	Linear interpolation
`ADX_ALGO_IM_LIX`	Linear Interpolation with extrapolation for out of bound values
`ADX_ALGO_IM_LOG`	Loglinear interpolation

ADX_ALGO_IM_STEP	Step interpolation without extrapolation
ADX_ALGO_IM_STEPX	Step interpolation with extrapolation for out of bound values
ADX_ALGO_IM_TSPLINE	T-spline interploation without extrapolation
ADX_ALGO_IM_TSPLINEX	T-spline interploation with extrapolation for out of bound values
ADX_ALGO_IM_VOL	Linear interpolation on the square value of the points (for volatilities)

See also

- "AdxAlgorithmYesNo" on page 347

AdxAlgorithmYesNo

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxAlgorithmYesNo")]
enum AdxAlgorithmYesNo
 {
 ADX_ALGO_YN_NO                                    =2,
 ADX_ALGO_YN_YES                                   =1,
 } AdxAlgorithmYesNo;
```

The AdxAlgorithmYesNo enumeration specifies whether interpolation is used or not.

Values

ADX_ALGO_YN_NO	Interpolation is not used
ADX_ALGO_YN_YES	Interpolation is used

See also

- "AdxAlgorithmInterp" on page 346

AdxAODMT

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxAODMT")]
enum AdxAODMT
 {
 ADX_AODMT_DEFAULT                                 =1,
 ADX_AODMT_EXACT                                   =2,
 ADX_AODMT_RIEMANN                                 =3
 } AdxAODMT;
```

The AdxAODMT enumeration specifies whether an exact or approximate integration method is used in the pricing of the CDS.

Values

ADX_AODMT_DEFAULT	The default is Riemann
ADX_AODMT_EXACT	The integrals are calculated exactly
ADX_AODMT_RIEMANN	The integrals are calculated with a numerical approximation

See also

- "AdxAttrNToDefaultCDS" on page 324

AdxApproxType

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxApproxType")]
enum AdxApproxType
{
ADX_APPROX_MIDDLE =1,
ADX_APPROX_RIGHT =2,
ADX_APPROX_VOL =3
} AdxApproxType;
```

The `AdxApproxType` enumeration specifies the approximation type for calculation methods.

Values

ADX_APPROX_MIDDLE	The approximation type is Middle
ADX_APPROX_RIGHT	The approximation type is Right
ADX_APPROX_VOL	The approximation type is Volatility

See also

"AdxAttrCalcMethod" on page 301

AdxAsianType

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxAsianType")]
enum AdxAsianType
{
ADX_ASIAN_NONE =3,
ADX_ASIAN_RATE =2,
ADX_ASIAN_STRIKE =1
} AdxAsianType;
```

The `AdxAsianType` enumeration specifies the type of Asian option.

Values

ADX_ASIAN_NONE	No option type is specified
ADX_ASIAN_RATE	Average rate (price) option
ADX_ASIAN_STRIKE	Average strike option

See also

"AdxAsian" on page 229

AdxAssetType

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxAssetType")]
enum AdxAssetType
 {
 ADX_ASSETTYPE_UI_COM                                    =1,
 ADX_ASSETTYPE_UI_CUR                                    =2,
 ADX_ASSETTYPE_UI_FUT                                    =4,
 ADX_ASSETTYPE_UI_IR                                     =5,
 ADX_ASSETTYPE_UI_NOVALUE                                =6,
 ADX_ASSETTYPE_UI_SEC                                    =3
 } AdxAssetType;
```

The `AdxAssetType` enumeration specifies asset type.

Values

ADX_ASSETTYPE_UI_COM	Commodities
ADX_ASSETTYPE_UI_CUR	Currencies
ADX_ASSETTYPE_UI_EQ	Equities
ADX_ASSETTYPE_UI_FUT	Futures
ADX_ASSETTYPE_UI_IR	Interest rate based assets
ADX_ASSETTYPE_UI_NOVALUE	No asset type is specified
ADX_ASSETTYPE_UI_SEC	Securities

See also

- "AdxAsset" on page 230

AdxAverageType

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxAverageType")]
enum AdxAverageType
```

```
{
ADX_AVE_ARI                                    =1,
ADX_AVE_GEO                                    =2,
ADX_AVE_NONE                                   =3
} AdxAverageType;
```

The `AdxAverageType` enumeration specifies the average type.

Values

`ADX_AVE_ARI`	Arithmetic average
`ADX_AVE_GEO`	Geometric average
`ADX_AVE_NONE`	No value for AVE Keyword

AdxBinaryType

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxBinaryType")]
enum AdxBinaryType
{
ADX_BINARY_ASSET                               =2,
ADX_BINARY_CASH                                =1
} AdxBinaryType;
```

The `AdxBinaryType` enumeration specifies the type of binary option.

Values

`ADX_BINARY_ASSET`	Binary option to specify an asset-or-nothing option
`ADX_BINARY_CASH`	Binary option to specify a cash-or-nothing option

See also

- "AdxOpBinary" on page 270

AdxBSModelType

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxBSModelType")]
enum AdxBSModelType
{
ADX_BSMODEL_CONT                               =2,
ADX_BSMODEL_REG                                =3,
ADX_BSMODEL_STEP                               =1
} AdxBSModelType;
```

The AdxBSModelType enumeration specifies the Black and Scholes Model type.

Values

ADX_BSMODEL_CONT

ADX_BSMODEL_REG

ADX_BSMODEL_STEP

AdxBsVol

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxBsVol")]
enum AdxBsVol
{
ADX_BSVOL_ALLIN                                    =2,
ADX_BSVOL_CAPLET                                   =1
} AdxBsVol;
```

The AdxBsVol enumeration specifies which volatility Adfin Analytics calculates internally from the input surface to use in the valuation of the cap or of the derivatives.

Values

ADX_BSVOL_ALLIN	Return the all-in volatilities
ADX_BSVOL_CAPLET	Return the caplet volatilities

AdxCalibrationType

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxCalibrationType")]
enum AdxCalibrationType
{
ADX_CALIBRATION_DIRECT                             =0,
ADX_CALIBRATION_MSF                                =1
} AdxCalibrationType;
```

The AdxCalibrationType enumeration specifies calibration type.

Values

ADX_CALIBRATION_DIRECT

ADX_CALIBRATION_MSF

AdxCallPutType

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxCallPutType")]
enum AdxCallPutType
 {
 ADX_CALLPUT_CALL                                    =1,
 ADX_CALLPUT_MIXED                                   =4,
 ADX_CALLPUT_NONE                                    =2,
 ADX_CALLPUT_PUT                                     =-1
 } AdxCallPutType;
```

The `AdxCallPutType` enumeration specifies call or put option type.

Values

`ADX_CALLPUT_CALL`	Call option
`ADX_CALLPUT_MIXED`	Both Call and Put option
`ADX_CALLPUT_NONE`	No value specified
`ADX_CALLPUT_PUT`	Put option

AdxCapFloorType

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxCapFloorType")]
enum AdxCapFloorType
 {
 ADX_CPF_AUTOFIXCAP                                  =5,
 ADX_CPF_AUTOFIXFLOOR                                =6,
 ADX_CPF_CAP                                         =2,
 ADX_CPF_CAPFLOOR                                    =4,
 ADX_CPF_CMS_SPREADOPTION                            =13,
 ADX_CPF_FLOOR                                       =3,
 ADX_CPF_NO                                          =1,
 ADX_CPF_RATCHETCAP                                  =7,
 ADX_CPF_RATCHETFLR                                  =8,
 ADX_CPF_STICKYCAP                                   =9,
 ADX_CPF_STICKYFLR                                   =10,
 ADX_FLXCAP                                          =11,
 ADX_FLXFLOOR                                        =12
 } AdxCapFloorType;
```

The AdxCapFloorType enumeration specifies cap or floor type.

Values

ADX_CPF_AUTOFIXCAP	Auto-flexi cap
ADX_CPF_AUTOFIXFLOOR	Auto-flexi floor
ADX_CPF_CAP	
ADX_CPF_CAPFLOOR	
ADX_CPF_CMS_SPREADOPTION	CMS Spread Option
ADX_CPF_FLOOR	
ADX_CPF_NO	
ADX_CPF_RATCHETCAP	Ratchet-cap
ADX_CPF_RATCHETFLR	Ratchet-floor
ADX_CPF_STICKYCAP	Sticky-cap
ADX_CPF_STICKYFLR	Sticky-floor
ADX_FLXCAP	Flexi cap (bermuda type)
ADX_FLXFLOOR	Flexi floor (bermuda type)

See also

- "AdxCapFloor" on page 239

AdxCDOType

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxCDOType")]
enum AdxCDOType
 {
 ADX_CDO_CASH                                    =1,
 ADX_CDO_SYNTHETIC                               =2
 } AdxCDOType;
```

The AdxCDOType enumeration specifies CDO type.

Values

ADX_CDO_CASH	Cash CDO
ADX_CDO_SYNTHETIC	Synthetic CDO

See also

- "AdxCDOTranche" on page 244

AdxCLDRADJ

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxCLDRADJ")]
enum AdxCLDRADJ
 {
 ADX_CLDRADJ_CLDR                                    =3,
 ADX_CLDRADJ_NO                                      =1,
 ADX_CLDRADJ_NULL                                    =5,
 ADX_CLDRADJ_WEEKEND                                 =4,
 ADX_CLDRADJ_YES                                     =2
 } AdxCLDRADJ;
```

The AdxCLDRADJ enumeration specifies calendar adjustment type.

Values

ADX_CLDRADJ_CLDR	Calculate the coupon date with calendar
ADX_CLDRADJ_NO	Calculate the coupon date without any calendar adjustment
ADX_CLDRADJ_NULL	Calculate the coupon date with null calendar
ADX_CLDRADJ_WEEKEND	Calculate the coupon date with calendar weekend
ADX_CLDRADJ_YES	Calculate the coupon date with default settings calendar

AdxCouponCalculationMethod

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxCouponCalculationMethod")]
enum AdxCouponCalculationMethod
 {
 ADX_CCM_00                                          =24,
 ADX_CCM_05G                                         =31,
 ADX_CCM_0AG                                         =30,
 ADX_CCM_A0                                          =22,
 ADX_CCM_A0D                                         =10,
 ADX_CCM_A0NL                                        =25,
 ADX_CCM_A4                                          =37,
 ADX_CCM_A5                                          =21,
 ADX_CCM_A5D                                         =8,
 ADX_CCM_A5NL                                        =13,
 ADX_CCM_A5P                                         =28,
 ADX_CCM_AA                                          =20,
```

```
ADX_CCM_AA5                    =9,
ADX_CCM_AIZ                    =39,
ADX_CCM_BB00                   =4,
ADX_CCM_BB0M                   =6,
ADX_CCM_BBA0                   =3,
ADX_CCM_BBA4                   =27,
ADX_CCM_BBA5                   =2,
ADX_CCM_BBA5JD                 =43,
ADX_CCM_BBAA                   =1,
ADX_CCM_BBAAI                  =33,
ADX_CCM_BBAAX                  =32,
ADX_CCM_BBE0                   =5,
ADX_CCM_BBEM                   =7,
ADX_CCM_BBITL                  =19,
ADX_CCM_BBW252                 =38,
ADX_CCM_CST                    =34,
ADX_CCM_DISC                   =40,
ADX_CCM_FRF                    =23,
ADX_CCM_E0                     =23,
ADX_CCM_FRF                    =18,
ADX_CCM_G0                     =29,
ADX_CCM_IT                     =26,
ADX_CCM_IT2                    =36,
ADX_CCM_ITL                    =19,
ADX_CCM_JAP                    =11
ADX_CCM_JAPDEB                 =35
ADX_CCM_JAPMUN                 =41,
ADX_CCM_JAPNTT                 =42,
ADX_CCM_MM00                   =16,
ADX_CCM_MMA0                   =14,
ADX_CCM_MMA5                   =15,
ADX_CCM_MMAA                   =17,
ADX_CCM_MME0                   =26,
ADX_CCM_MMNL0                  =25,
ADX_CCM_MMNL5                  =13,
ADX_CCM_W252                   =12
} AdxCouponCalculationMethod;
```

The `AdxCouponCalculationMethod` enumeration specifies the coupon calculation method.

Values

`ADX_CCM_00`	00 30/360
`ADX_CCM_05G`	05G 30/365 German
`ADX_CCM_0AG`	0AG 30/Actual German
`ADX_CCM_A0`	A0 Actual/360
`ADX_CCM_A0D`	A0D Actual/360 Day Based
`ADX_CCM_A0NL`	A0NL (Actual - leap Day) / 360
`ADX_CCM_A4`	A4 Kenyan accrued Actual/364
`ADX_CCM_A5`	A5 Actual/365
`ADX_CCM_A5D`	A5D Actual/365 Day Based
`ADX_CCM_A5NL`	A5NL (Actual - leap Day) / 365
`ADX_CCM_A5P`	A5P Actual/365 proportionate
`ADX_CCM_AA`	AA Actual/Actual
`ADX_CCM_AA5`	AA5 Actual/365.25
`ADX_CCM_AIZ`	AIZ Russian accrued (Act+1)/Act
`ADX_CCM_BB00`	BB00 Bond 30/360
`ADX_CCM_BB0M`	BB0M Bond Basis 30/360 (US) Modified
`ADX_CCM_BBA0`	BBA0 Bond Actual/360
`ADX_CCM_BBA4`	BBA4 Kenyan Bond Actual/364
`ADX_CCM_BBA5`	BBA5 Bond Actual/365
`ADX_CCM_BBA5JD`	BBA5JD Bond Actual/365 (6 months) with broken period computed from Date D date and 365 japan basis
`ADX_CCM_BBAA`	BBAA Bond Actual/Actual
`ADX_CCM_BBAAI`	BBAAI Bond Actual/Actual with broken period computed frm issue date
`ADX_CCM_BBAAX`	BBAAX Bond Actual/Actual with specific broken adjusted period computed (Eurex compatible)
`ADX_CCM_BBE0`	BBE0 Bond 30E/360 ISMA
`ADX_CCM_BBEM`	BBEM Bond Basis 30E/360 AIBD Modified
`ADX_CCM_BBITL`	BBITL Bond Italian
`ADX_CCM_BBW252`	BBW252 Brazilian bond basis W252
`ADX_CCM_CST`	CST Constant : no period calculation taken into account
`ADX_CCM_DISC`	DISC China Zero Coupon Accrued
`ADX_CCM_E0`	E0 30E/360 ISMA
`ADX_CCM_FRF`	FRFTMP French TMP, T4M, TAM

ADX_CCM_G0	G0 30/360 German
ADX_CCM_IT	IT 30E+1/360
ADX_CCM_IT2	IT2
ADX_CCM_ITL	ITL Italian first Coupon
ADX_CCM_JAP	JAP Japan : (A5 or A5 + 1)
ADX_CCM_JAPDEB	JAPNGDEBENTURES Non Government Japan bonds, bank debentures
ADX_CCM_JAPMUN	JAPNGMUNICIPAL Non Government Japan municipal and corporate bonds
ADX_CCM_JAPNTT	JAPNGNTT Non Government Japan bonds, NTT Bonds
ADX_CCM_MM00	MM00 Money Market 30/360
ADX_CCM_MMA0	MMA0 MoneyMarket : Actual/360 - Number of days
ADX_CCM_MMA5	MMA5 MoneyMarket : Actual/365 - Number of days
ADX_CCM_MMAA	MMAA Money Market Actual/Actual
ADX_CCM_MME0	MME0 Money Market 30E/360 ISMA
ADX_CCM_MMNL0	MMNL0 (Actual - Leap day) / 360
ADX_CCM_MMNL5	MMNL5 (Actual - Leap day) / 365
ADX_CCM_W252	W252 252 Working days

AdxDateMovingConvention

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxDateMovingConvention")]
enum AdxDateMovingConvention
 {
 ADX_DMC_FOLLOWING                                    = 3,
 ADX_DMC_MODIFIEDFOLLOWING                            = 4
 ADX_DMC_NO                                           = 1,
 ADX_DMC_PRECEDING                                    = 2,
 ADX_DMC_THIRDWEDNESDAY                               =5
 } AdxDateMovingConvention;
```

The `AdxDateMovingConvention` enumeration defines the set of values, which can be used to adjust cashflow dates of objects derived from the `IAdxLeg` interface.

Values

`ADX_DMC_FOLLOWING` Moves the date to the following working day

ADX_DMC_MODIFIEDFOLLOWING	Moves the date to the following working day unless it pushes the date into the next month. In this case, the last working day of the month is used
ADX_DMC_NO	Does not move the date to the preceding working day
ADX_DMC_PRECEDING	Moves the date to the preceding working day
ADX_DMC_THIRDWEDNESDAY	For moving the date to the third wednesday of the month (or next working day if third wednesday is not a working day)

See also

- "AdxBarrierCapFloor" on page 230
- "AdxBond" on page 235
- "AdxCapFloor" on page 239
- "AdxConvBond" on page 247
- "AdxDigitalCapFloor" on page 254
- "AdxEndOfMonthConvention" on page 363
- "AdxFixedLeg" on page 257
- "AdxFloatLeg" on page 257
- "AdxFrn" on page 260
- "AdxIlb" on page 266
- "IAdxLeg Interface" on page 205

AdxDaysOffset

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxDaysOffset")]
enum AdxDaysOffset
 {
 ADX_OFFSET_MAX                                    =3,
 ADX_OFFSET_NONE                                   =0,
 ADX_OFFSET_ONE                                    =1,
 ADX_OFFSET_TWO                                    =2
 } AdxDaysOffset;
```

The AdxDaysOffset enumeration defines the trading date offset.

Values

ADX_OFFSET_MAX	Maximum working days
ADX_OFFSET_NONE	No working days
ADX_OFFSET_ONE	One working day
ADX_OFFSET_TWO	Two working days

AdxDcbType

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxDcbType")]
```

```
enum AdxDcbType
 {
 ADX_DCB_CM_00                                    =4,
 ADX_DCB_CM_00M                                   =6,
 ADX_DCB_CM_05                                    =15,
 ADX_DCB_CM_05G                                   =19,
 ADX_DCB_CM_0AG                                   =18,
 ADX_DCB_CM_A0                                    =3,
 ADX_DCB_CM_A0D                                   =10,
 ADX_DCB_CM_A0NL                                  =14,
 ADX_DCB_CM_A4                                    =16,
 ADX_DCB_CM_A5                                    =2,
 ADX_DCB_CM_A5D                                   =8,
 ADX_DCB_CM_A5NL                                  =13,
 ADX_DCB_CM_AA                                    =1,
 ADX_DCB_CM_AA5                                   =9,
 ADX_DCB_CM_BBA5JD                                =22,
 ADX_DCB_CM_BBAAI                                 =20,
 ADX_DCB_CM_BBAAX                                 =21,
 ADX_DCB_CM_E0                                    =5,
 ADX_DCB_CM_E0M                                   =7,
 ADX_DCB_CM_G0                                    =17,
 ADX_DCB_CM_JAP                                   =11,
 ADX_DCB_CM_NOTYPE                                =0,
 ADX_DCB_CM_W252                                  =12
 } AdxDcbType;
```

The `AdxDcbType` enumeration defines the day count basis.

Values

`ADX_DCB_CM_00`	30/360 (US)
`ADX_DCB_CM_00M`	30/360 (US) Modified
`ADX_DCB_CM_05`	30/365 (Brazil)
`ADX_DCB_CM_05G`	30/365 German
`ADX_DCB_CM_0AG`	30/Actual German
`ADX_DCB_CM_A0`	Actual/360
`ADX_DCB_CM_A0D`	Actual/360 - Number of Days
`ADX_DCB_CM_A0NL`	(Actual - Leap day) / 360

ADX_DCB_CM_A4	Actual/364 (Kenyan)
ADX_DCB_CM_A5	Actual/365
ADX_DCB_CM_A5D	Actual/365 - Number of Days
ADX_DCB_CM_A5NL	(Actual - Leap day) / 365
ADX_DCB_CM_AA	Actual/Actual
ADX_DCB_CM_AA5	Actual/365.25 - Number of Day
ADX_DCB_CM_BBA5JD	Bond Actual/365 (6 months) with broken period computed from Date D date and 365 Japan basis
ADX_DCB_CM_BBAAI	Actual/Actual (from Issue)
ADX_DCB_CM_BBAAX	Actual/Actual (for Eurex)
ADX_DCB_CM_E0	30E/360 AIBD
ADX_DCB_CM_E0M	30E/360 AIBD Modified
ADX_DCB_CM_G0	30/360 German
ADX_DCB_CM_JAP	Actual/365, discard the 29 February when over 1 year
ADX_DCB_CM_NOTYPE	No type defined
ADX_DCB_CM_W252	Actual number of business day / 252

See also

- "AdxBarrierCapFloor" on page 230
- "AdxBond" on page 235
- "AdxCapFloor" on page 239
- "AdxConvBond" on page 247
- "AdxDigitalCapFloor" on page 254
- "AdxEndOfMonthConvention" on page 363
- "AdxFixedLeg" on page 257
- "AdxFloatLeg" on page 257
- "AdxFrn" on page 260
- "AdxIlb" on page 266

AdxDCP

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxDCP")]
enum AdxDCP
 {
 ADX_DCP_NO                                    =1,
 ADX_DCP_YES                                   =2
 } AdxDCP;
```

The AdxDCP enumeration defines the current payment parameter for cap or floor functions.

Values

ADX_DCP_NO	Keep the current caplet, or floorlet
ADX_DCP_YES	Skip the current caplet, or floorlet

See also

- "AdxDigitalCapFloor" on page 254
- "IAdxDigitalCapFloor Interface" on page 194
- "AdxAttrCalcMethod" on page 301

AdxDilutionFlag

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxDilutionFlag")]
enum AdxDilutionFlag
 {
 ADX_DILUTION_NO                                                    =2,
 ADX_DILUTION_YES                                                   =1
 } AdxDilutionFlag;
```

The AdxDilutionFlag enumeration defines whether dilution is taken into account for warrants.

Values

ADX_DILUTION_NO	Ignore dilution
ADX_DILUTION_YES	Take dilution into account

See also

- "AdxOption" on page 275
- "IAdxOption Interface" on page 209

AdxDividendCstGrowthType

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxDividendCstGrowthType")]
enum AdxDividendCstGrowthType
 {
 ADX_DIVIDEND_CSTGROWTH_ESTIMATED                                   =2,
 ADX_DIVIDEND_CSTGROWTH_HIST                                        =1
 } AdxDividendCstGrowthType;
```

The AdxDividendCstGrowthType enumeration defines whether to use the estimated or historical dividend.

Values

ADX_DIVIDEND_CSTGROWTH_ESTIMATED	Estimated dividend
ADX_DIVIDEND_CSTGROWTH_HIST	Historical dividend

See also

- "AdxDividendModel" on page 257
- "IAdxDividendModel Interface" on page 195

AdxDividendType

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxDividendType")]
enum AdxDividendType
 {
 ADX_DIV_FIXED_DISCOUNT                                    =3,
 ADX_DIV_FIXED_JUMP                                        =2,
 ADX_DIV_PROP_PERCENT                                      =4,
 ADX_DIV_YES                                               =5,
 ADX_DIV_YIELD                                             =1
 } AdxDividendType;
```

The `AdxDividendType` enumeration defines the dividend type.

Values

ADX_DIV_FIXED_DISCOUNT	Fixed discounted dividend
ADX_DIV_FIXED_JUMP	Fixed jump dividend
ADX_DIV_PROP_PERCENT	Proportional (in percent) dividend
ADX_DIV_YES	
ADX_DIV_YIELD	Continuous dividend yield

See also

- "AdxDividendModel" on page 257
- "IAdxDividendModel Interface" on page 195

AdxDivUserDefined

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxDivUserDefined")]
enum AdxDivUserDefined
 {
 ADX_DIVIDEND_USERDEFINED_NO                               =2,
```

```
ADX_DIVIDEND_USERDEFINED_UNDEFINED =3,
ADX_DIVIDEND_USERDEFINED_YES =1
} AdxDivUserDefined;
```

The `AdxDivUserDefined` enumeration defines whether a user-defined model is used for calculating the dividend.

Values

`ADX_DIVIDEND_USERDEFINED_NO`	User-defined model is not used
`ADX_DIVIDEND_USERDEFINED_UNDEFINED`	Not specified whether a user-defined model is used or not
`ADX_DIVIDEND_USERDEFINED_YES`	User-defined model is used

See also

- "AdxDividendModel" on page 257
- "IAdxDividendModel Interface" on page 195

AdxEndOfMonthConvention

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxEndOfMonthConvention")]
enum AdxEndOfMonthConvention
{
ADX_EMC_DEFAULT = 3,
ADX_EMC_LAST = 1,
ADX_EMC_SAME = 2,
ADX_EMC_SAME28 =4
} AdxEndOfMonthConvention;
```

The `AdxEndOfMonthConvention` enumeration defines the set of values, which can be used to adjust cashflow dates of objects derived from the IAdxLeg interface.

Values

`ADX_EMC_DEFAULT`	Sets the calculated date according to the default value
`ADX_EMC_LAST`	Sets the calculated date to the last working day
`ADX_EMC_SAME`	Sets the calculated date to the same day. In this latter case, the date may be moved according to the date moving convention if it is a non-working day
`ADX_EMC_SAME28`	Sets the calculated date to the same day, 28FEB being always considered as the last working day

See also

- "AdxBarrierCapFloor" on page 230

- "AdxBond" on page 235
- "AdxCapFloor" on page 239
- "AdxConvBond" on page 247
- "AdxDateMovingConvention" on page 357
- "AdxDigitalCapFloor" on page 254
- "AdxFixedLeg" on page 257
- "AdxFloatLeg" on page 257
- "AdxFrn" on page 260
- "AdxIlb" on page 266
- "IAdxLeg Interface" on page 205

AdxErrorMode

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 2.0 AdxErrorMode")]
enum MRV_ErrorMode
 {
 DIALOGBOX                                    = 2,
 EXCEPTION                                    = 3,
 NO_EXCEPTION                                 = 1,
 } AdxErrorMode;
```

The `AdxErrorMode` enumeration defines the set of values, which can be assigned to the `ErrorMode` property of each of the Analytics' objects.

You can set the `ErrorMode` property to define the behavior of the object when an error condition arises. According to the value of this property, either an exception is raised (`EXCEPTION`), a dialog box is displayed indicating that an error has occurred (`DIALOGBOX`), or nothing is done (`NO_EXCEPTION`), in which case your code must check for errors.

Values

`DIALOG_BOX`	An error encountered by the object opens a dialog box displaying the error that has occurred
`EXCEPTION`	An error encountered by the object will cause an exception to occur in the program, interrupting the normal flow of control
`NO_EXCEPTION`	An error encountered by the object will not cause an interruption in the program. In this mode, the client application must check the value of the `ErrorCode` property to determine whether an error has been encountered

See also

- "IAdxObject Interface: ErrorMode" on page 168

AdxExerciseMode

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxExerciseMode")]
enum AdxExerciseMode
```

```
{
ADX_EXM_AMER                                    =1,
ADX_EXM_BERM                                    =3,
ADX_EXM_EURO                                    =2,
ADX_EXM_NOVALUE                                 =0
} AdxExerciseMode;
```

The `AdxExerciseMode` enumeration defines the exercise type.

Values

`ADX_EXM_AMER`	Specifies a vanilla option with an American mode
`ADX_EXM_BERM`	Specifies a vanilla option with a Bermudan mode
`ADX_EXM_EURO`	Specifies a European option
`ADX_EXM_NOVALUE`	No exercise mode is specified

See also

- "AdxOption" on page 275
- "AdxBarrierCapFloor" on page 230
- "AdxDigitalCapFloor" on page 254
- "IAdxOption Interface" on page 209
- "IAdxBarrierCapFloor Interface" on page 181
- "IAdxDigitalCapFloor Interface" on page 194

AdxExerciseStrategy

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxExerciseStrategy")]
enum AdxExerciseStrategy
{
ADX_ES_1                                        =1,
ADX_ES_2                                        =2,
ADX_ES_3                                        =3,
ADX_ES_4                                        =4,
ADX_ES_NOVALUE                                  =5
} AdxExerciseStrategy;
```

The `AdxExerciseStrategy` enumeration defines the option exercise strategy.

Values

`ADX_ES_1`

`ADX_ES_2`

```
ADX_ES_3
ADX_ES_4
ADX_ES_NOVALUE
```

See also

- "AdxOption" on page 275
- "AdxBarrierCapFloor" on page 230
- "AdxDigitalCapFloor" on page 254
- "IAdxOption Interface" on page 209
- "IAdxBarrierCapFloor Interface" on page 181
- "IAdxDigitalCapFloor Interface" on page 194

AdxFrequency

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxFrequency")]
enum AdxFrequency
{
ADX_FRQ_180D5                    =180,
ADX_FRQ_182D                     =182,
ADX_FRQ_183D                     =183,
ADX_FRQ_1D                       =365,
ADX_FRQ_28D                      =28,
ADX_FRQ_364D                     =364,
ADX_FRQ_365D                     =360,
ADX_FRQ_90D                      =90,
ADX_FRQ_91D                      =91,
ADX_FRQ_92D                      =92,
ADX_FRQ_DEFAULT                  =367,
ADX_FRQ_EY                       =368,
ADX_FRQ_MONTHLY                  =12,
ADX_FRQ_QUARTERLY                =4,
ADX_FRQ_R2                       =185,
ADX_FRQ_R4                       =95,
ADX_FRQ_SEMIANNUAL               =2,
ADX_FRQ_USERCODE                 =369,
ADX_FRQ_YEARLY                   =1,
ADX_FRQ_ZERO                     =366
} AdxFrequency;
```

The `AdxFrequency` enumeration defines the compounding frequency.

Values

ADX_FRQ_180D5	For setting the Compounding Frequency semi-annual (but every 180 days)
ADX_FRQ_182D	For setting the Compounding Frequency every 182 days
ADX_FRQ_183D	For setting the Compounding Frequency every 183 days
ADX_FRQ_1D	For setting the Compounding Frequency every day
ADX_FRQ_28D	For setting the Compounding Frequency every 28 days
ADX_FRQ_364D	For setting the Compounding Frequency every 364 days
ADX_FRQ_365D	For setting the Compounding Frequency every 365 days
ADX_FRQ_90D	For setting the Compounding Frequency every 90 days
ADX_FRQ_91D	For setting the Compounding Frequency every 91 days
ADX_FRQ_92D	For setting the Compounding Frequency every 92 days
ADX_FRQ_DEFAULT	For setting the Compounding Frequency to the Default value
ADX_FRQ_EY	For setting the Compounding Frequency to the Equivalent Yield value
ADX_FRQ_MONTHLY	For setting the Compounding Frequency to monthly
ADX_FRQ_QUARTERLY	For setting the Compounding Frequency to quarterly
ADX_FRQ_R2	For setting the Compounding Frequency semi-annual of 182 and 183 days
ADX_FRQ_R4	For setting the Compounding Frequency quarterly of 91 days, 91 days, 91 days and 92 days
ADX_FRQ_SEMIANNUAL	For setting the Compounding Frequency to semi-annual
ADX_FRQ_USERCODE	For setting the Compounding Frequency to a custom value through the Adfin Settings Manager
ADX_FRQ_YEARLY	For setting the Compounding Frequency to yearly
ADX_FRQ_ZERO	For setting no Compounding Frequency

See also

- "AdxBond" on page 235
- "AdxConvBond" on page 247
- "AdxFuture" on page 263
- "AdxIIb" on page 266
- "AdxOption" on page 275
- "AdxRepo" on page 282
- "AdxSwap" on page 289
- "AdxTermStructure" on page 292
- "AdxFrq" on page 263

AdxFrequencyType

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxFrequencyType")]
```

```
enum AdxFrequencyType
 {
 ADX_FREQ_DAY                                    =252,
 ADX_FREQ_DEFAULT                                =367,
 ADX_FREQ_MONTH                                  =12,
 ADX_FREQ_WEEK                                   =52,
 ADX_FREQ_YEAR                                   =1,
 ADX_FREQ_ZERO                                   =365
 } AdxFrequencyType;
```

The `AdxFrequencyType` enumeration defines the frequency type.

Values

`ADX_FREQ_DAY`	For setting the Frequency to daily (252 observations)
`ADX_FREQ_DEFAULT`	For setting the Frequency to the default value
`ADX_FREQ_MONTH`	For setting the Frequency to monthly
`ADX_FREQ_WEEK`	For setting the Frequency to weekly
`ADX_FREQ_YEAR`	For setting the Frequency to yearly
`ADX_FREQ_ZERO`	For setting the Frequency to daily (365 observations)

See also

- "AdxFrq" on page 263

AdxFrom

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxFrom")]
enum AdxFrom
 {
 ADX_FROM_SETTLE                                 =2,
 ADX_FROM_TRADE                                  =1
 } AdxFrom;
```

The `AdxForm` enumeration defines type of date input used for calculations.

Values

`ADX_FROM_SETTLE`	Use the settlement date as input
`ADX_FROM_TRADE`	Use the trade date as input

See also

- "AdxBond" on page 235
- "AdxConvBond" on page 247

- "AdxFrn" on page 260

AdxFTType

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxFTType")]
enum AdxFTType
 {
 ADX_FT_BRIGO                                    =12,
 ADX_FT_BS                                       =2,
 ADX_FT_CEV                                      =10,
 ADX_FT_UNDEF                                    =1,
 ADX_FT_WHA                                      =8
 } AdxFTType;
```

The `AdxFTType` enumeration defines option formula type.

Values

`ADX_FT_BRIGO`	Use the Brigo model
`ADX_FT_BS`	Use the Black and Scholes model
`ADX_FT_CEV`	Use the Constant Elasticity of Variance model
`ADX_FT_UNDEF`	Model is not defined
`ADX_FT_WHA`	Use the Whaley model

See also

- "AdxOption" on page 275
- "IAdxOption Interface" on page 209

AdxFutureReferenceRule

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxFutureReferenceRule")]
enum AdxFutureReferenceRule
 {
 ADX_FUTURE_REFERENCE_RULE_10C                   =2,
 ADX_FUTURE_REFERENCE_RULE_14C                   =11,
 ADX_FUTURE_REFERENCE_RULE_15C                   =3,
 ADX_FUTURE_REFERENCE_RULE_1C                    =1,
 ADX_FUTURE_REFERENCE_RULE_20C                   =4,
 ADX_FUTURE_REFERENCE_RULE_25CPrevious           =8,
 ADX_FUTURE_REFERENCE_RULE_2FRI                  =6,
```

```
ADX_FUTURE_REFERENCE_RULE_3WED                    =5,
ADX_FUTURE_REFERENCE_RULE_Brent                   =9,
ADX_FUTURE_REFERENCE_RULE_LASTWDPrevious          =10,
ADX_FUTURE_REFERENCE_RULE_NBB                     =7
} AdxFutureReferenceRule;
```

The `AdxFutureReferenceRule` enumeration defines the futures contract reference date calculation method.

Values

`ADX_FUTURE_REFERENCE_RULE_10C`	Set the date to the 10th calendar day of the delivery month
`ADX_FUTURE_REFERENCE_RULE_14C`	Set the date to the 14th calendar day of the delivery month
`ADX_FUTURE_REFERENCE_RULE_15C`	Set the date to the 15th calendar day of the delivery month
`ADX_FUTURE_REFERENCE_RULE_1C`	Set the date to the first calendar day of the delivery month
`ADX_FUTURE_REFERENCE_RULE_20C`	Set the date to the 20th calendar day of the delivery month
`ADX_FUTURE_REFERENCE_RULE_25CPrevious`	Set the date to the 25th calendar day of the month prior to the delivery month
`ADX_FUTURE_REFERENCE_RULE_2FRI`	Set the date to the second Friday of the delivery month
`ADX_FUTURE_REFERENCE_RULE_3WED`	Set the date to the third Wednesday of the delivery month
`ADX_FUTURE_REFERENCE_RULE_Brent`	Set the date to 15 calendar days prior to the first day of the delivery month for a brent future contract
`ADX_FUTURE_REFERENCE_RULE_LASTWDPrevious`	Set the date to the last working day of the for contracts such as natural gas
`ADX_FUTURE_REFERENCE_RULE_NBB`	Set the date to the third Wednesday after the ninth day of the contract month (specific to the NZ Bank Bill Future Contracts)

See also

- "AdxFuture" on page 263
- "IAdxFuture Interface" on page 201

AdxIC

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxIC")]
enum AdxIC
{
ADX_IC_LONG                                       =5,
```

```
ADX_IC_LONGR                                    =6,
ADX_IC_NBC                                      =7,
ADX_IC_REGULAR                                  =1,
ADX_IC_SHORT                                    =2,
ADX_IC_SHORTP                                   =3,
ADX_IC_SHORTR                                   =4
} AdxIC;
```

The `AdxIC` enumeration defines the irregular first coupon type.

Values

`ADX_IC_LONG`	For long first coupon (first coupon date equal to second anniversary date)
`ADX_IC_LONGR`	For long first coupon with regular nominal value and starting accrued date equal to first anniversary date
`ADX_IC_NBC`	For NBC first coupon
`ADX_IC_REGULAR`	For regular first coupon
`ADX_IC_SHORT`	For short first coupon (first coupon date equal to first anniversary date)
`ADX_IC_SHORTP`	For short first coupon with proportional value
`ADX_IC_SHORTR`	For short first coupon with regular nominal value

See also

- "AdxBond" on page 235
- "AdxFrn" on page 260
- "AdxSwap" on page 289
- "IAdxBond Interface" on page 183
- "IAdxFrn Interface" on page 199
- "IAdxSwap Interface" on page 220

AdxICF

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxICF")]
enum AdxICF
{
ADX_ICF_ALL                                     =1,
ADX_ICF_IN                                      =5,
ADX_ICF_IO                                      =2,
ADX_ICF_NONE                                    =3,
ADX_ICF_PO                                      =4
```

```
} AdxICF;
```

The `AdxICF` enumeration defines the index cashflows.

Values

`ADX_ICF_ALL`	Indexed cashflows on both principal and interest
`ADX_ICF_IN`	Indexed cashflows on interest only
`ADX_ICF_IO`	Indexed cashflows on interest only
`ADX_ICF_NONE`	Interest and principal are not indexed
`ADX_ICF_PO`	Indexed cashflows on principal only

See also

- "AdxStyle" on page 226
- "IAdxStyle Interface" on page 215

AdxICM

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxICM")]
enum AdxICM
{
ADX_ICM_AUSTRALIA                                    =3,
ADX_ICM_BRL                                          =5,
ADX_ICM_GILT                                         =6,
ADX_ICM_INTERP                                       =1,
ADX_ICM_MOSTO                                        =4,
ADX_ICM_POLAND                                       =7,
ADX_ICM_PREVIOUS                                     =2
} AdxICM;
```

The `AdxICM` enumeration defines the daily inflation reference and coupon calculation method.

Values

`ADX_ICM_AUSTRALIA`	For Australian index-linked bonds
`ADX_ICM_BRL`	For Brazilian index-linked bonds
`ADX_ICM_GILT`	For UK government index-linked bonds
`ADX_ICM_INTERP`	For Canadian, French, Swedish and US index-linked bonds
`ADX_ICM_MOSTO`	For Russian index-linked bonds
`ADX_ICM_POLAND`	For Polish index-linked bonds
`ADX_ICM_PREVIOUS`	For UK index-linked bonds

See also

- "AdxIIb" on page 266
- "IAdxIIb Interface" on page 204
- "AdxStyle" on page 226
- "IAdxStyle Interface" on page 215

AdxIndexDate

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxIndexDate")]
enum AdxIndexDate
 {
 ADX_INDEX_ENDDATE                                    =2,
 ADX_INDEX_STARTDATE                                  =1
 } AdxIndexDate;
```

The `AdxIndexDate` enumeration defines the start and end date for the index period.

Values

`ADX_INDEX_ENDDATE`	Period end date
`ADX_INDEX_STARTDATE`	Period start date

See also

- "AdxIIb" on page 266
- "IAdxIIb Interface" on page 204
- "AdxStyle" on page 226
- "IAdxStyle Interface" on page 215

AdxInstrumentType

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxInstrumentType")]
enum AdxInstrumentType
 {
 ADX_INSTTYPE_BOND                                    =3,
 ADX_INSTTYPE_CDS                                     =1,
 ADX_INSTTYPE_DF                                      =2
 } AdxInstrumentType;
```

The `AdxInstrumentType` enumeration defines the credit model calibration method.

Values

`ADX_INSTTYPE_BOND`	Calibrates the model by using risky bond prices

ADX_INSTTYPE_CDS	Calibrates the model by using a credit default swap curve
ADX_INSTTYPE_DF	Calibrates the model by using a credit zero-coupon curve

See also

- "AdfinX Analytics Objects" on page 228
- "AdfinX Analytics Interfaces" on page 167
- "AdxAttrTermStructure" on page 337

AdxInterceptYesNo

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxInterceptYesNo")]
enum AdxInterceptYesNo
{
ADX_INTERCEPT_NO                                    =1,
ADX_INTERCEPT_YES                                   =2
} AdxInterceptYesNo;
```

The AdxInterceptYesNo enumeration specifies whether or not there is an intercept in the regression.

Values

ADX_INTERCEPT_NO	There is an intercept in the regression
ADX_INTERCEPT_YES	There is no intercept in the regression

AdxIOType

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxIOType")]
enum AdxIOType
{
ADX_IO_CASH                 =1,
ADX_IO_PERCENT              =2
} AdxIOType;
```

The AdxIOType enumeration defines the format of function inputs and outputs.

Values

ADX_IO_CASH	Determines that inputs and outputs are expressed in their current currency
ADX_IO_PERCENT	Determines that inputs and outputs are expressed as a percentage of the face value

See also

- "AdxConvBond" on page 247
- "AdxConvBond" on page 247

AdxIrsPvbpMethod

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxIrsPvbpMethod")]
enum AdxIrsPvbpMethod
{
ADX_IRS_CURVE                                    =1,
ADX_IRS_PARALLEL                                 =2,
ADX_IRS_SUM                                      =3
} AdxIrsPvbpMethod;
```

The `AdxIrsPvbpMethod` enumeration calculates the price value of a basis point of an interest rate swap.

Values

`ADX_IRS_CURVE`	Calculates partial sensitivities
`ADX_IRS_PARALLEL`	Calculates global difference in net present values for a parallel shift in the yield curve by one basis point
`ADX_IRS_SUM`	Calculates the sum of all partial sensitivities

See also

- "AdxSwap" on page 289
- "IAdxSwap Interface" on page 220

AdxLayOut

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxLayOut")]
enum AdxLayOut
{
ADX_LAY_HORIZ                                    = 1,
ADX_LAY_VER                                      = 2
} AdxLayOut ;
```

The `AdxLayOut` enumeration defines the way the array of results can be displayed.

Values

`ADX_LAY_HORIZ`	Display an array with an horizontal array layout
`ADX_LAY_VER`	Display an array with a vertical layout

AdxLegAttr

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxLegAttr")]
enum AdxLegAttr
 {
 ADX_LEG_BOTH                                    =3,
 ADX_LEG_FIXED                                   =4,
 ADX_LEG_FLOAT                                   =5,
 ADX_LEG_PAID                                    =1,
 ADX_LEG_RANGE_ACCRUAL_NOTE                      =6,
 ADX_LEG_RECEIVED                                =2
 } AdxLegAttr ;
```

The `AdxLegAttr` enumeration specifies the current leg type of a swap.

Values

`ADX_LEG_BOTH`	
`ADX_LEG_FIXED`	Current leg is a fixed leg
`ADX_LEG_FLOAT`	Current leg is a floating leg
`ADX_LEG_PAID`	Current leg is a paid leg
`ADX_LEG_RANGE_ACCRUAL_NOTE`	Current leg is a range accrual leg
`ADX_LEG_RECEIVED`	Current leg is a received leg

See also

- "AdxSwap" on page 289
- "IAdxSwap Interface" on page 220

AdxLookBackType

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxLookBackType")]
enum AdxLookBackType
 {
 ADX_LOOK_LADDER                                 =3,
 ADX_LOOK_NONE                                   =4,
 ADX_LOOK_SPOT                                   =1,
 ADX_LOOK_STRIKE                                 =2
 } AdxLookBackType ;
```

The `AdxLookBackType` enumeration specifies the current leg type of .

Values

ADX_LOOK_LADDER	Ladder option (Not a value of LOOK Keyword)
ADX_LOOK_NONE	No value for LOOK Keyword
ADX_LOOK_SPOT	StrikePrice = UnderlyingMinMax
ADX_LOOK_STRIKE	Fixed StrikePrice

See also

- "AdxOpLookBack" on page 272
- "IAdxOpLookBack Interface" on page 208

AdxNormalType

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxNormalType")]
enum AdxNormalType
{
ADX_NORMAL_230                =1,
ADX_NORMAL_250                =2,
ADX_NORMAL_400                =3,
ADX_NORMAL_ABROMOVITZ         =5,
ADX_NORMAL_ERF                =8,
ADX_NORMAL_HART               =6,
ADX_NORMAL_MARSAGLIA          =7,
ADX_NORMAL_NEW                =4
} AdxNormalType ;
```

The AdxNormalType enumeration specifies the approximations used for the normal distribution function.

Values

ADX_NORMAL_230	
ADX_NORMAL_250	
ADX_NORMAL_400	
ADX_NORMAL_ABROMOVITZ	Uses the Abramowitz formula
ADX_NORMAL_ERF	Uses the error function
ADX_NORMAL_HART	Uses the Hart formula
ADX_NORMAL_MARSAGLIA	Uses the Marsaglia formula
ADX_NORMAL_NEW	

AdxOriginPeriod

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxOriginPeriod")]
enum AdxOriginPeriod
 {
 ADX_FROM_FXSPOT                                    =256,
 ADX_FROM_FXTRADE                                   =512,
 ADX_FROM_MMSPOT                                    =768,
 ADX_FROM_MMTRADE                                   =1024
 } AdxOriginPeriod ;
```

The `AdxOriginPeriod` enumeration specifies the date calculation origin for forex and money markets.

Values

`ADX_FROM_FXSPOT`	Uses the spot date as origin and the Forex market spot offset
`ADX_FROM_FXTRADE`	Uses the trading date as origin and the Forex market spot offset
`ADX_FROM_MMSPOT`	Uses the spot date as origin and the Money market spot offset
`ADX_FROM_MMTRADE`	Uses the trading date as origin and the Money market spot offset

See also

- "AdxForex" on page 258
- "IAdxForex Interface" on page 197

AdxPayment

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxPayment")]
enum AdxPayment
 {
 ADX_PAYMENT_END                                    =2,
 ADX_PAYMENT_START                                  =1
 } AdxPayment ;
```

The `AdxPayment` enumeration specifies the payment date date.

Values

`ADX_PAYMENT_END`	Payment ends
`ADX_PAYMENT_START`	Payment starts

AdxPEX

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxPEX")]
```

```
enum AdxPEX
 {
 ADX_PEX_BOTH                                                    =1,
 ADX_PEX_END                                                     =2,
 ADX_PEX_NONE                                                    =3,
 ADX_PEX_START                                                   =4
 } AdxPEX ;
```

The `AdxPEX` enumeration specifies regarding the exchange of notional principal cashflows.

Values

`ADX_PEX_BOTH`	All principal cashflows are exchanged
`ADX_PEX_END`	Principal cashflows are exchanged except the first one
`ADX_PEX_NONE`	No principal cashflows are exchanged
`ADX_PEX_START`	Only first principal cashflow is exchanged

See also

- "AdxSwap" on page 289
- "IAdxSwap Interface" on page 220

AdxPxType

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxPxType")]
enum AdxPxType
 {
 ADX_PRICE_CLEAN                                                 =2,
 ADX_PRICE_GROSS                                                 =1
 } AdxPxType ;
```

The `AdxPxType` enumeration defines the price type.

Values

`ADX_PRICE_CLEAN`	Clean price
`ADX_PRICE_GROSS`	Gross price

See also

- "AdxBond" on page 235
- "AdxConvBond" on page 247
- "AdxFrn" on page 260
- "AdxIlb" on page 266
- "IAdxBond Interface" on page 183

- "IAdxConvBond Interface" on page 190
- "IAdxFrn Interface" on page 199
- "IAdxIlb Interface" on page 204

AdxQuotationMode

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxQuotationMode")]
enum AdxQuotationMode
 {
 ADX_QM_CROSS                    =4,
 ADX_QM_DIRECT                   =1,
 ADX_QM_INDIRECT                 =2,
 ADX_QM_USD_CROSS                =3
 } AdxQuotationMode ;
```

The `AdxQuotationMode` enumeration defines the currency quotation mode.

Values

`ADX_QM_CROSS`	
`ADX_QM_DIRECT`	Direct mode
`ADX_QM_INDIRECT`	Indirect mode
`ADX_QM_USD_CROSS`	

See also

- "AdxCrossCurrency" on page 251
- "AdxForex" on page 258
- "IAdxCrossCurrency Interface" on page 192
- "IAdxForex Interface" on page 197

AdxRateModelType

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxRateModelType")]
enum AdxRateModelType
 {
 ADX_RATEMODEL_BDT               =7,
 ADX_RATEMODEL_BGM               =10,
 ADX_RATEMODEL_BS                =9,
 ADX_RATEMODEL_CEVLMM            =11,
 ADX_RATEMODEL_CFR               =16,
```

```
ADX_RATEMODEL_DDLMM                =18,
ADX_RATEMODEL_HJM                  =17,
ADX_RATEMODEL_HW                   =8,
ADX_RATEMODEL_NOVALUE              =0,
ADX_RATEMODEL_SPLINE               =12,
ADX_RATEMODEL_VF                   =6,
ADX_RATEMODEL_VFM                  =13,
ADX_RATEMODEL_YTA                  =2,
ADX_RATEMODEL_YTB                  =3,
ADX_RATEMODEL_YTC                  =14,
ADX_RATEMODEL_YTM                  =1,
ADX_RATEMODEL_YTP                  =15,
ADX_RATEMODEL_YTW                  =4,
ADX_RATEMODEL_ZC                   =5
} AdxRateModelType ;
```

The `AdxRateModelType` enumeration defines the rate model type.

Values

`ADX_RATEMODEL_BDT`	For Black, Derman, and Toy model
`ADX_RATEMODEL_BGM`	
`ADX_RATEMODEL_BS`	For Black and Scholes model
`ADX_RATEMODEL_CEVLMM`	
`ADX_RATEMODEL_CFR`	For stepwise constant forward rate models
`ADX_RATEMODEL_DDLMM`	
`ADX_RATEMODEL_HJM`	
`ADX_RATEMODEL_HW`	For Hull and White model
`ADX_RATEMODEL_NOVALUE`	No value
`ADX_RATEMODEL_SPLINE`	For spline model
`ADX_RATEMODEL_VF`	For Vasicek-Fong model
`ADX_RATEMODEL_VFM`	
`ADX_RATEMODEL_YTA`	To adapt the calculation of the yield to the bond structure
`ADX_RATEMODEL_YTB`	For yield to best
`ADX_RATEMODEL_YTC`	For yield to call (when no date is entered take the next call)
`ADX_RATEMODEL_YTM`	For yield to maturity
`ADX_RATEMODEL_YTP`	For yield to put (when no date is entered take the next put)
`ADX_RATEMODEL_YTW`	For yield to worst
`ADX_RATEMODEL_ZC`	

See also

- "AdxBond" on page 235
- "AdxIlb" on page 266
- "AdfinX Analytics Interfaces" on page 167
- "AdxSwap" on page 289
- "IAdxIlb Interface" on page 204
- "IAdxFrn Interface" on page 199
- "IAdxSwap Interface" on page 220

AdxRateOfReturn

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxRateOfReturn")]
enum AdxRateOfReturn
 {
 ADX_RATEOFRETURN_ABS_LN                =3,
 ADX_RATEOFRETURN_ABS_LOG10             =2,
 ADX_RATEOFRETURN_ABSOLUTE              =1,
 ADX_RATEOFRETURN_RATIO                 =4,
 ADX_RATEOFRETURN_RATIO_LN              =6,
 ADX_RATEOFRETURN_RATIO_LOG10           =5
 } AdxRateOfReturn ;
```

The `AdxRateOfReturn` enumeration defines how to convert input values to a specific rate for equities regression analysis.

Values

`ADX_RATEOFRETURN_ABS_LN`	Absolute value of natural log
`ADX_RATEOFRETURN_ABS_LOG10`	Absolute value of log to the base 10
`ADX_RATEOFRETURN_ABSOLUTE`	Absolute log value
`ADX_RATEOFRETURN_RATIO`	Ratio log value
`ADX_RATEOFRETURN_RATIO_LN`	Ratio value of natural log
`ADX_RATEOFRETURN_RATIO_LOG10`	Ratio value of log to the base 10

AdxRateType

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxRateType")]
enum AdxRateType
 {
 ADX_RT_ACTUAL                          =1,
 ADX_RT_ACTUAL_BMA2003                  =13,
```

```
ADX_RT_CMPJAP                    =6,
ADX_RT_CONSTANT                  =14,
ADX_RT_CONTINUOUS                =2,
ADX_RT_DISCOUNT                  =4,
ADX_RT_FREQUENCY                 =-1,
ADX_RT_LFT                       =12,
ADX_RT_MMB                       =7,
ADX_RT_MMM                       =9,
ADX_RT_MMP                       =8,
ADX_RT_MMR                       =10,
ADX_RT_MONEYMARKET               =3,
ADX_RT_NOVALUE                   =0,
ADX_RT_SIMPLEJAP                 =5,
ADX_RT_TRE                       =11
} AdxRateType ;
```

The `AdxRateType` enumeration defines the rate type.

Values

`ADX_RT_ACTUAL`	
`ADX_RT_ACTUAL_BMA2003`	
`ADX_RT_CMPJAP`	For compounded yield/rate
`ADX_RT_CONSTANT`	
`ADX_RT_CONTINUOUS`	For continuous yield/rate
`ADX_RT_DISCOUNT`	For discounted yield/rate
`ADX_RT_FREQUENCY`	
`ADX_RT_LFT`	For Brazilian LFT yield method
`ADX_RT_MMB`	Specified when using the Money Market Bullet pricing method
`ADX_RT_MMM`	Specified when using the Money Market Medium pricing method
`ADX_RT_MMP`	Specified when using the Money Market Proceeds pricing method
`ADX_RT_MMR`	Specified when using the Money Market Direct Discounting pricing method
`ADX_RT_MONEYMARKET`	For Money Market yield/rate
`ADX_RT_NOVALUE`	No value specified
`ADX_RT_SIMPLEJAP`	For simple yield/rate
`ADX_RT_TRE`	For US Bills Treasury

See also

- "AdxBond" on page 235
- "AdxConvBond" on page 247
- "AdxIlb" on page 266
- "AdxRepo" on page 282
- "IAdxBond Interface" on page 183
- "IAdxConvBond Interface" on page 190
- "IAdxIlb Interface" on page 204
- "IAdxRepo Interface" on page 211

AdxReferenceDate

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxReferenceDate")]
enum AdxReferenceDate
 {
 ADX_REFDATE_ISSUE                                          =1,
 ADX_REFDATE_MATURITY                                       =2,
 ADX_REFDATE_NonAdjustedMaturity_PivotFirstBrokenDate       =3
 } AdxReferenceDate ;
```

The `AdxReferenceDate` defines the reference date in cash flow generation.

Values

`ADX_REFDATE_ISSUE`	Uses the issue date as reference date
`ADX_REFDATE_MATURITY`	Uses the maturity date as reference date
`ADX_REFDATE_NonAdjustedMaturity_PivotFirstBrokenDate`	

See also

- "AdxBond" on page 235
- "AdxConvBond" on page 247
- "AdxIlb" on page 266
- "AdxCDOTranche" on page 244
- "IAdxBond Interface" on page 183
- "IAdxConvBond Interface" on page 190
- "IAdxIlb Interface" on page 204

AdxReset

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxReset")]
enum AdxReset
```

```
{
ADX_RESET_ADVANCE                    =1,
ADX_RESET_ARREARS                    =2,
ADX_RESET_UNDEFINED                  =3
} AdxReset ;
```

The `AdxReset` defines when the forward rate is reset. It is used for pricing LIBOR in arrears swaps.

Values

`ADX_RESET_ADVANCE`	Specifies that the forward rate is reset in advance
`ADX_RESET_ARREARS`	Specifies that the forward rate is reset in arrears
`ADX_RESET_UNDEFINED`	Forward rate is not defined

See also

- "AdxSwap" on page 289
- "IAdxSwap Interface" on page 220

AdxRiskModelType

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxRiskModelType")]
enum AdxRiskModelType
{
ADX_RISKMODEL_CIR                    =1,
ADX_RISKMODEL_CURVE                  =2,
ADX_RISKMODEL_MULTI                  =4,
ADX_RISKMODEL_POISSON                =3
} AdxRiskModelType ;
```

The `AdxRiskModelType` defines the credit model type.

Values

`ADX_RISKMODEL_CIR`	Indicates that the credit model is provided by the Cox, Ingersoll, and Ross model
`ADX_RISKMODEL_CURVE`	Indicates that the credit model is provided by the credit event probability curve
`ADX_RISKMODEL_MULTI`	Indicates that the credit model is provided by a copula model with {i} constituents
`ADX_RISKMODEL_POISSON`	Indicates that the credit model is provided by the exponential poisson model

See also

- "AdxCDOTranche" on page 244
- "AdxNToDefaultCDS" on page 269

AdxRoundingMode

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxRoundingMode")]
enum AdxRoundingMode
 {
 ADX_RND_DOWN                                                    =1,
 ADX_RND_NEAR                                                    =3,
 ADX_RND_UP                                                      =2
 } AdxRoundingMode ;
```

The `AdxRoundingMode` defines the rounding mode.

Values

`ADX_RND_DOWN`	Round the number down
`ADX_RND_NEAR`	Round the number to the nearest possible number (depends on the tick)
`ADX_RND_UP`	Round the number up

AdxRT

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxRT")]
enum AdxRT
 {
 ADX_RT_BULLET                                                   =1,
 ADX_RT_CONSTANT_ANNUITIES                                       =11,
 ADX_RT_EQUALSERIES                                              =2,
 ADX_RT_EQUALSERIES_12PERIODS                                    =10,
 ADX_RT_EQUALSERIES_2PERIODS                                     =3,
 ADX_RT_EQUALSERIES_3PERIODS                                     =4,
 ADX_RT_EQUALSERIES_4PERIODS                                     =5,
 ADX_RT_EQUALSERIES_5PERIODS                                     =6,
 ADX_RT_EQUALSERIES_6PERIODS                                     =7,
 ADX_RT_EQUALSERIES_7PERIODS                                     =8,
 ADX_RT_EQUALSERIES_8PERIODS                                     =9,
 ADX_RT_PERPETUAL                                                =13,
```

```
ADX_RT_SCHEDULE                                    =12
} AdxRT ;
```

The `AdxRT` defines the reimbursement type.

Values

`ADX_RT_BULLET`	For bullet or in fine
`ADX_RT_CONSTANT_ANNUITIES`	For constant annuities
`ADX_RT_EQUALSERIES`	
`ADX_RT_EQUALSERIES_12PERIODS`	
`ADX_RT_EQUALSERIES_2PERIODS`	
`ADX_RT_EQUALSERIES_3PERIODS`	
`ADX_RT_EQUALSERIES_4PERIODS`	
`ADX_RT_EQUALSERIES_5PERIODS`	
`ADX_RT_EQUALSERIES_6PERIODS`	
`ADX_RT_EQUALSERIES_7PERIODS`	
`ADX_RT_EQUALSERIES_8PERIODS`	
`ADX_RT_PERPETUAL`	For perpetual bonds
`ADX_RT_SCHEDULE`	

See also

- "AdxBond" on page 235
- "AdxFrn" on page 260
- "IAdxBond Interface" on page 183
- "IAdxFrn Interface" on page 199

AdxSolverFrom

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxSolverFrom")]
enum AdxSolverFrom
{
ADX_SOLVER_FROM_BPV                                =7,
ADX_SOLVER_FROM_CONV                               =8,
ADX_SOLVER_FROM_DELTA                              =2,
ADX_SOLVER_FROM_GAMMA                              =3,
ADX_SOLVER_FROM_PREMIUM                            =1,
ADX_SOLVER_FROM_RHO                                =4,
ADX_SOLVER_FROM_THETA                              =6,
ADX_SOLVER_FROM_VEGA                               =5
} AdxSolverFrom ;
```

The AdxSolverFrom specifies the input type to define the calculation methods available for pricing instruments.

Values

ADX_SOLVER_FROM_BPV	From basis point value
ADX_SOLVER_FROM_CONV	From convexity
ADX_SOLVER_FROM_DELTA	From delta
ADX_SOLVER_FROM_GAMMA	From gamma
ADX_SOLVER_FROM_PREMIUM	From premium
ADX_SOLVER_FROM_RHO	From rho
ADX_SOLVER_FROM_THETA	From theta
ADX_SOLVER_FROM_VEGA	From vega

See also

- "AdxCalcMethod" on page 238
- "IAdxCalcMethod Interface" on page 172

AdxSortedDataAscDes

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxSortedDataAscDes")]
enum AdxSortedDataAscDes
 {
 ADX_SORTED_ASC                    =1,
 ADX_SORTED_DES                    =2
 } AdxSortedDataAscDes ;
```

The AdxSortedDataAscDes defines the sorting order.

Values

ADX_SORTED_ASC	Sorts in ascending order
ADX_SORTED_DES	Sorts in descending order

AdxSwapNpvType

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxSwapNpvType")]
enum AdxSwapNpvType
 {
 ADX_NPV_BOTH                                    =1,
 ADX_NPV_FIXED                                   =2,
 ADX_NPV_FLOAT                                   =3,
```

```
ADX_NPV_PAID                                    =4,
ADX_NPV_RECEIVED                                =5
} AdxSwapNpvType ;
```

The `AdxSwapNpvType` defines the current leg of the swap.

Values

`ADX_NPV_BOTH`	
`ADX_NPV_FIXED`	Current leg is a fixed leg
`ADX_NPV_FLOAT`	Current leg is a floating leg
`ADX_NPV_PAID`	Current leg is a paid leg
`ADX_NPV_RECEIVED`	Current leg is a received leg

See also

- "AdxSwap" on page 289
- "IAdxSwap Interface" on page 220

AdxSwapType

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxSwapType")]
enum AdxSwapType
{
ADX_CDS_AMER                    =1,
ADX_CDS_EURO                    =2,
ADX_CDS_UNDEFINED               =5,
ADX_TRS_BOND                    =3,
ADX_TRS_EQUITY                  =4
} AdxSwapType ;
```

The `AdxSwapType` defines the swap type.

Values

`ADX_CDS_AMER`

`ADX_CDS_EURO`

`ADX_CDS_UNDEFINED`

`ADX_TRS_BOND`

`ADX_TRS_EQUITY`

See also

- "AdxSwap" on page 289
- "IAdxSwap Interface" on page 220

AdxTaxProRata

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxTaxProRata")]
enum AdxTaxProRata
 {
 ADX_TPR_IT_BOT                    =4,
 ADX_TPR_IT_BTP                    =3,
 ADX_TPR_IT_CTZ                    =2,
 ADX_TPR_NO                        =1
 } AdxTaxProRata ;
```

The `AdxTaxProRata` defines pro rata tax type.

Values

`ADX_TPR_IT_BOT`	For Italian BOT
`ADX_TPR_IT_BTP`	For Italian BTP
`ADX_TPR_IT_CTZ`	For Italian CTZ
`ADX_TPR_NO`	No pro rata tax is specified

See also

- "AdxBond" on page 235
- "IAdxBond Interface" on page 183

AdxTouch

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxTouch")]
enum AdxTouch
 {
 ADX_TOUCH_DEFERRED                =3,
 ADX_TOUCH_NO                      =2,
 ADX_TOUCH_ONE                     =1,
 ADX_TOUCH_UNDEFINED               =4
 } AdxTouch ;
```

The `AdxTouch` defines the binary path dependent option type.

Values

`ADX_TOUCH_DEFERRED`	For a deferred one-touch option
`ADX_TOUCH_NO`	For a no-touch option
`ADX_TOUCH_ONE`	For a one-touch option with immediate payment
`ADX_TOUCH_UNDEFINED`	Not defined

See also

- "AdxOption" on page 275
- "IAdxOption Interface" on page 209

AdxTransfMethod

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxTransfMethod")]
enum AdxTransfMethod
 {
 ADX_TRANSF_JPM                    =3,
 ADX_TRANSF_NOMETHOD               =1,
 ADX_TRANSF_POISSON                =2,
 } AdxTransfMethod ;
```

The `AdxTransfMethod` specifies the forward default intensity.

Values

`ADX_TRANSF_JPM`	Corresponds to IMPROBA:CFI. Constant Forward Intensity (i.e. JPM methodology)
`ADX_TRANSF_NOMETHOD`	Corresponds to IMPROBA:LIN. Linear interpolation on the default probas
`ADX_TRANSF_POISSON`	Corresponds to IMPROBA:LAI. Linear interpolation on the average intensity

See also

- "AdxNToDefaultCDS" on page 269

AdxVolType

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxVolType")]
enum AdxVolType
 {
 ADX_VOL_SR                =1,
 ADX_VOL_ZC                =2,
 } AdxVolType ;
```

The `AdxVolType` specifies the volatility type used in the dynamic model.

Values

`ADX_VOL_SR`	For short rates volatility
`ADX_VOL_ZC`	For zero -coupon yield volatility

See also

- "AdxVolatilityModel" on page 294
- "IAdxVolatilityModel Interface" on page 223

AdxWorkingDayConvention

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0
AdxWorkingDayConvention")]
enum AdxWorkingDayConvention
 {
 ADX_WDC_STARTEXCLUDED          =2,
 ADX_WDC_STARTINCLUDED          =1
 } AdxWorkingDayConvention ;
```

The `AdxWorkingDayConvention` specifies the working day convention.

Values

`ADX_WDC_STARTEXCLUDED`	Working Day: Start Day Excluded - End Date Included
`ADX_WDC_STARTINCLUDED`	Working Day: Start Day Included - End Date Excluded

AdxYesNo

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxYesNo")]
enum AdxYesNo
 {
 ADX_YN_NO                          =2,
 ADX_YN_YES                         =1
 } AdxYesNo;
```

The `AdxYesNo` specifies the working day convention.

Values

```
ADX_YN_NO
ADX_YN_YES
```

AdxZcType

```
typedef [v1_enum, uuid(...), helpstring("AdfinX Analytics 3.0 AdxZcType")]
enum AdxZcType
 {
 ADX_ZCTYPE_DF                      =1,
 ADX_ZCTYPE_RATE                    =2
```

```
} AdxZcType;
```

The `AdxZcType` defines the curve type.

Values

`ADX_ZCTYPE_DF`	Discount factor
`ADX_ZCTYPE_RATE`	Rates