CLIB user guide
Version 1.100.0.0

Sartorial Programming

26 October 2020

Contents

Chapter 1

Introduction

This is an example SPI library containing wrapper functions for the ISDA/CDS C-library.

In general we will not be using C++ except for error handling, hence the name CLIB (C-library)

Chapter 2

Data types

Each item of data in the library is of a particular data type. Arbitrarily complex data types can be built up of simpler types, with ultimately everything being broken down into the fundamental built-in types.

When describing the inputs to a function or data fields in a class we will use the term `attribute` indicating the data type and name of the input or class data field. Attributes in addition can be an array or a matrix. We will indicate a simple array by the notation `[]` after the data type. We will indicate a matrix by the notation `[,]` after the data type. Note that a matrix is a rectangular array where each row in the array has the same number of elements.

In this section we describe all the possible data types available in the library, starting from the built-in types and building up to the classes available in the library.

2.1 Built-in types

Built-in types are data types that are provided as standard for all services. These types are the building blocks of all data representations.

Typename	**Description**
bool	True or false.
int	Positive or negative integer (32-bits).
double	Double precision real number (64-bits).
string	Standard string type.
Date	Date type. This will be translated into the appropriate type for each context. For Excel, this is the number of days since 1900. For Python, this is the `datetime.date` class. For .NET this is the `System.DateTime` class.
char	Single character.
unsigned int	Integer that will always be positive

2.2 Simple types

Simple types are types which are represented by one of the standard types in the public interface (bool, int, double, string, Date, char, unsigned int).

However internally to the library the type is converted to some other type understood by the library itself.

This means that the user sees instances of the standard type which are then converted (and possibly validated) to the type understood by the library.

2.2.1 Holidays

Holidays is represented by string in the public interface.

Holidays in the system are represented by strings.

The string either represents a filename (which we read once only for efficiency), or a holiday name created directly, or NONE (no holidays, weekends are not working days), or No_Weekends (no holidays, weekends are working days).

2.3 Enumerated types

Enumerated types are types which can take one of a finite number of possible values. At the C++ level, these are represented by an enum inside a class. The possible values hence are given the namespace of the class name which corresponds to the name defined in the documentation.

For the Excel interface, the values that can be used are strings. These will then be converted into the C++ enumerated type, and then used within the low-level library code.

In the documentation, we will give the name of the enumerated type, followed by a general description of how the type is used, and then followed by a list of the possible values (known as the enumerands).

2.3.1 BadDayConv

Bad day convention.

This is used to determine rules for adjusting bad days (holidays and weekends) to good days (working days).

Possible Values:

Code	Strings	Description
FOLLOW	Follow, F	
PREVIOUS	Previous, P	
NONE	None, N	
MODIFIED	ModifiedFollow, M	

2.3.2 CashOrSwap

No description.

Possible Values:

Code	Strings	Description
IGNORE	Ignore, I, ""	
CASH	Cash, C	
SWAP	Swap, S	

2.3.3 InterpType

Interpolation type for extracting rates from curves.

Possible Values:

Code	Strings	Description
LINEAR_INTERP	LINEAR_INTERP	Linear interpolation.
FLAT_FORWARDS	FLAT_FORWARDS	Flat forwards interpolation.

2.3.4 IntervalAdjType

No description.

Possible Values:

Code	Strings	Description
CALENDAR	CALENDAR, C	
BUSINESS	BUSINESS, B	
WEEKDAY	WEEKDAY, W	

2.3.5 Math.BinaryOperator

Defines a binary operation.

Possible Values:

Code	Strings	Description
Add	+	Adds two number together
Subtract	-	Subtracts the second number from the first number
Multiply	*	Multiplies two numbers together
Divide	/	Divides the first number by the second number
Power	**	Raises the first number to the power of the second number

2.3.6 Weekday

No description.

Possible Values:

Code	Strings	Description
MONDAY	Monday, Mon	
TUESDAY	Tuesday, Tue	
WEDNESDAY	Wednesday, Wed	
THURSDAY	Thursday, Thu	
FRIDAY	Friday, Fri	
SATURDAY	Saturday, Sat	
SUNDAY	Sunday, Sun	

2.4 Classes

Classes contain data represented by attributes and can have function represented by so-called class methods.

Attributes will have a name and data type, and contain data of the defined data type. Attributes can be scalars or arrays. The data type can be any previously defined data type, e.g another class, or a simple type, or an enumerated type, or one of the built-in primitive data types.

Classes can in general be serialised. This can be to a string format, or written to file in various formats, or transmitted via the network when using a client-server model.

In general, all of the attributes will be needed when we serialise the class, but not all of the attributes are needed for general information about the class. Hence some attributes are marked as private (only used in serialisation), and others as public (can be accessed directly from outside the class). At present we do not support changing the value of an attribute of class, but this may change in the future.

Classes can also have class methods. In the C++ interface these are C++ member functions. In the Excel interface class methods appear as functions where the name of the function starts with the class name followed by the method name. The first parameter of the function is the class instance (or object). In other interfaces to come later we will generally use the natural representation for each language supported. For example, Python and .NET both support classes, so we will have equivalent classes in these languages corresponding to the C++ class.

In the C++ interface we will use a form of shared pointer for representing the class. The normal constructors will be protected and you need to use the static `Make` method for each class in order to create the shared pointer instance for the class. By protecting the constructor in this manner, we make it impossible for client code to use raw pointers, and this should make it easier to avoid memory leaks in client C++ code.

2.4.1 CashFlowList

Stores dates and amounts of cash flows. The date/amount pairs are not necessarily stored in date order.

Attributes:

Type and Name	Description
Date dates[]	Array of dates. The dates do not have to ordered.
double amounts[]	Array of cash amounts. Each amount must correspond to a date in the dates array at the same index. The number of amounts must be the same as the number of dates.

Properties:

Type and Name	Description
int numItems	Number of dates/amounts.

Note that properties can be accessed in the same way as attributes, but are not part of the constructor or the serialization of the class.

Methods:

Prototype	Description
operator DateList()	Converts to DateList. Coerces to a DateList using the dates of the cash flow list

Constructors:

- CashFlowListMerge (see page 27)
- SwapFixedFlows (see page 36)

2.4.2 CashSwapCurve

No description.

Attributes:

Type and Name	Description
Date baseDate	
CashOrSwap cashOrSwaps[]	See CashOrSwap (page 8).
Date dates[]	
double rates[]	
DayCountConv cashDcc	See DayCountConv (page 18).
DateInterval swapFixedInterval	See DateInterval (page 13).
DayCountConv swapFixedDcc	See DayCountConv (page 18).
BadDayConv badDayConv	See BadDayConv (page 8).
Holidays holidays	See Holidays (page 7).

Methods:

Prototype	Description
CashOrSwap[] CashOrSwaps()	See page 41 for further details.
ZeroCurve (see page 24) ZeroCurve(InterpType, string)	See page 41 for further details.

2.4.3 DateInterval

No description.

Attributes:

Type and Name	Description
string name	

Properties:

Type and Name	Description
char units	
int periods	

Note that properties can be accessed in the same way as attributes, but are not part of the constructor or the serialization of the class.

Methods:

Prototype	Description
static DateInterval (see page 13) Make(int, char)	Constructs DateInterval from number of periods and period type. See page 41 for further details.

Coercion:

Instances of DateInterval can be coerced from other types as follows:

Type and Name	Description
string name	

2.4.4 DateList

No description.

Attributes:

Type and Name	Description
Date dates[]	

Properties:

Type and Name	Description
int numItems	

Note that properties can be accessed in the same way as attributes, but are not part of the constructor or the serialization of the class.

Methods: None

Constructors:

- DateListMerge (see page 29)

2.4.5 DateOrInterval

No description.

Derived Classes:

- DateOrIntervalDate (see page 16)
- DateOrIntervalInterval (see page 17)

Attributes: None

Methods: None

Coercion:

Instances of DateOrInterval can be coerced from other types as follows:

Type and Name	Description
Date date	
string str	

2.4.6 DateOrIntervalDate

No description.

Base Class: DateOrInterval (see page 15)

Attributes:

Type and Name	Description
Date date	
BadDayConv badDayConv	See BadDayConv (page 8).

Methods: None

2.4.7 DateOrIntervalInterval

No description.

Base Class: DateOrInterval (see page 15)

Attributes:

Type and Name	Description
DateInterval ivl	See DateInterval (page 13).
BadDayConv badDayConv	Optional (default = `"F"`). See BadDayConv (page 8).
char adjType	Optional (default = `'C'`).

Methods: None

Constructors:

- DateOrIntervalIntervalMake (see page 30)

Coercion:

Instances of DateOrIntervalInterval can be coerced from other types as follows:

Type and Name	Description
string str	

2.4.8 DayCountConv

Day count convention.

This is used to compute fractions of a year between two dates.

Attributes:

Type and Name	Description
string name	

Properties:

Type and Name	Description
int id	

Note that properties can be accessed in the same way as attributes, but are not part of the constructor or the serialization of the class.

Methods: None

Coercion:

Instances of DayCountConv can be coerced from other types as follows:

Type and Name	Description
string name	
int id	

2.4.9 IndexCurve

No description.

Attributes:

Type and Name	Description
ZeroCurve zc	See ZeroCurve (page 24).
IndexCurveType curveType	See IndexCurveType (page 21).
IndexCurveFixings curveFixings	Optional. See IndexCurveFixings (page 20).

Methods: None

2.4.10 IndexCurveFixings

No description.

Attributes:

Type and Name	Description
Date dates[]	
double rates[]	

Methods: None

2.4.11 IndexCurveType

No description.

Attributes:

Type and Name	Description
string curveName	
DateInterval rateTenor	See DateInterval (page 13).
DayCountConv rateDayCountConv	See DayCountConv (page 18).
DateInterval fixedSwapInterval	See DateInterval (page 13).
DayCountConv fixedSwapDayCountConv	See DayCountConv (page 18).
DateInterval floatSwapInterval	See DateInterval (page 13).
DayCountConv floatSwapDayCountConv	See DayCountConv (page 18).
BadDayConv badDayConv	See BadDayConv (page 8).
Holidays holidayFile	See Holidays (page 7).

Methods: None

2.4.12 Math.Matrix

Defines a Matrix object consisting of a rectangular array of real numbers.

At present we don't provide any significant math functions which deal with the Matrix object.

Attributes:

Type and Name	Description
double data[,]	The data is a two-dimensional rectangular array of doubles.

Methods:

Prototype	Description
double[,] Data()	Returns all the data for the matrix. Should give the same result as accessing the data attribute. See page 42 for further details.
double Cell(int, int)	Returns an individual cell of the matrix. See page 41 for further details.
int Rows()	Returns the number of rows of the matrix. See page 42 for further details.
int Cols()	Returns the number of columns of the matrix. See page 42 for further details.
void Size()	See page 42 for further details.

2.4.13 RateType

Defines the type of the rate.

This can be a compound rate with a number of compounding periods per year, or a continuously compounded rate, or a simple rate.

Attributes:

Type and Name	Description
int number	

Properties:

Type and Name	Description
string name	

Note that properties can be accessed in the same way as attributes, but are not part of the constructor or the serialization of the class.

Methods:

Prototype	Description
static RateType (see page 23) Make(string)	See page 42 for further details.

Coercion:

Instances of RateType can be coerced from other types as follows:

Type and Name	Description
int number	Automatically converts an integer to a rate type. 1,2,4,12,365 = number of periods per year. 0 = simple rate (1/(1+rt)). 5000 = continuous basis. 512 = discount rate (1-rt). -2 = discount factor.
string name	

2.4.14 ZeroCurve

Defines a curve of zero coupon rates.

Attributes:

Type and Name	Description
Date baseDate	Base date of the curve. This is the date for which zero coupon prices are identically equal to 1.0.
InterpType interpType	Interpolation type used to interpolate rates for dates between the dates defined within the curve. See InterpType (page 8).
Date dates[]	Dates in the curve. These need to be unique and in ascending order.
double rates[]	Zero coupon rates in the curve with rates corresponding to dates.
RateType rateType	Describes how the rates are expressed, e.g. annually compounded, continously compounded etc. See RateType (page 23).
DayCountConv dayCountConv	Describes how we measure time in the curve. If we have a rate for time t then we must be able to calculate t for a given date. We do this using the day count convention. See DayCountConv (page 18).

Properties:

Type and Name	Description
int numItems	

Note that properties can be accessed in the same way as attributes, but are not part of the constructor or the serialization of the class.

Methods:

Prototype	Description
double ZeroRate(Date)	See page 43 for further details.
double PV(Date)	See page 43 for further details.
double FV(Date, Date)	See page 43 for further details.

Constructors:

- ZeroCurveCash (see page 39)
- ZeroCurveCashSwaps (see page 39)
- ZeroCurveSample (see page 39)
- ZeroCurveSwaps (see page 40)

Chapter 3

Functions and class methods

In this library there are two types of functions. We have free standing functions not attached to a particular class, and we have class methods which are attached to a class and for which you will need an instance of the class (also known as an Object) in order to make the call. In addition there are also so-called `static` methods of a class which behave more like free standing functions except that the name of the function also includes the class name.

Note that Excel does natively support the concept of class methods and only supports functions. Hence from within Excel when you have a class method you are actually calling a function with one extra parameter at the beginning of the argument list. The function name in this case includes the class name as well as the method name.

For Python and .NET which do support classes, then a class method is invoked by using the instance followed by `'.'` followed by the method name. Note that it is a consequence of the Python language that you can actually also use the Excel-style syntax for calling a class method. For `static` methods in Python and .NET then you need to use the class name instead of the instance when calling the `static` method.

Another difference between Excel and the other supported interfaces is that for Excel we represent Objects via string handles. For functions which create objects, there will be one extra parameter to the function (or method). This parameter is called `baseName` and is needed to provide the first part of the string which forms the Object handle.

3.1 Functions

These are free standing functions which are not attached to a particular class.

Note that when using the function you will need to use the overall namespace for the library as a prefix to the function name.

3.1.1 AddBusDays

No description.

Inputs:

Type and Name	Description
Date startDate	
int offset	
Holidays holidays	See Holidays (page 7).

ReturnType: Date

3.1.2 AddDateInterval

Adds a date interval to a given date.

Inputs:

Type and Name	Description
Date startDate	Start date.
DateInterval ivl	Date interval to add. See DateInterval (page 13).
int count	Optional (default = 1). Number of intervals to add (default = 1)

ReturnType: Date

3.1.3 BusDaysDiff

Calculates the number of business days between two dates (FROM & TO).

Algorithm: 1. if FROM = TO, the result is 0 2. if FROM < TO, the result is the number of business days in the CLOSED interval of [FROM+1,TO] 3. if FROM > TO, the result is negated number of business days in the CLOSED interval of [TO,FROM-1]

Inputs:

Type and Name	Description
Date fromDate	
Date toDate	
Holidays holidays	See Holidays (page 7).

ReturnType: int

3.1.4 BusinessDay

No description.

Inputs:

Type and Name	Description
Date date	
BadDayConv method	See BadDayConv (page 8).
Holidays holidays	See Holidays (page 7).

ReturnType: Date

3.1.5 BusinessEOM

Returns the last business day in the month of the given date

Inputs:

Type and Name	Description
Date date	Date which defines the current month
Holidays holidays	Name of the holiday file which defines which days are holidays See Holidays (page 7).

ReturnType: Date

3.1.6 CashFlowListMerge

Merges two cash flow lists together

Inputs:

Type and Name	Description
CashFlowList a	See CashFlowList (page 11).
CashFlowList b	See CashFlowList (page 11).

ReturnType: CashFlowList (see page 11)

3.1.7 ClockTime

Returns the clock time since the start of execution.

For Excel this function is volatile (which means that whenever you re-calculate the sheet the function will be re-calculated even though it has no dependencies) and hidden (which means it doesn't appear in the function wizard).

Inputs: None

ReturnType: double

3.1.8 ComputerName

No description.

Inputs: None

ReturnType: string

3.1.9 DOIEndDate

No description.

Inputs:

Type and Name	Description
Date startDate	
DateOrInterval doi	See DateOrInterval (page 15).
Holidays holidays	See Holidays (page 7).

ReturnType: Date

3.1.10 Date

Constructs a date from year, month and day using the Gregorian calendar.

Validates that the combination is valid.

Inputs:

Type and Name	Description
int year	Year A.D. - usually with 4 digits
int month	Month - 1 = January, 12 = December etc.
int day	Day of the month, from 1 to 31. It is an error to use a number more than the days in the particular month.

ReturnType: Date

3.1.11 DateFwdAdj

No description.

Inputs:

Type and Name	Description
Date startDate	
int days	
IntervalAdjType adjType	See IntervalAdjType (page 8).
Holidays holidays	See Holidays (page 7).

ReturnType: Date

3.1.12 DateFwdBus

Adds a number of date intervals to a given date and then adjusts the result so that it is a good business day.

Inputs:

Type and Name	Description
Date startDate	Start date.
int numIntervals	Number of intervals to move.
DateInterval ivl	Date interval to move. See DateInterval (page 13).
BadDayConv badDayMethod	Bad day convention. See BadDayConv (page 8).
Holidays holidays	Holiday file. See Holidays (page 7).

ReturnType: Date

3.1.13 DateIntervalFreq

Converts a date interval into a frequency of payments per year.

For example a date interval of 1M would correspond to 12 payments per year.

Inputs:

Type and Name	Description
DateInterval interval	See DateInterval (page 13).

ReturnType: double

3.1.14 DateIntervalYears

Converts a date interval into a number of years.

For a daily interval (e.g. 1D, 1W) this will be the number of days divided by 365. For a monthly interval (e.g. 1M, 1Y, 2Y) this will be the number of months divided by 12.

For example, 1M would give the answer 1/12 and 1W would give the answer 7/365.

Inputs:

Type and Name	Description
DateInterval interval	See DateInterval (page 13).

ReturnType: double

3.1.15 DateListMerge

No description.

Inputs:

Type and Name	Description
DateList dateList1	Optional. See DateList (page 14).
DateList dateList2	Optional. See DateList (page 14).

ReturnType: DateList (see page 14)

3.1.16 DateOrIntervalIntervalMake

No description.

Inputs:

Type and Name	Description
DateInterval ivl	See DateInterval (page 13).
BadDayConv badDayConv	Optional (default = `"F"`). See BadDayConv (page 8).
char adjType	Optional (default = `'C'`).

ReturnType: DateOrIntervalInterval (see page 17)

3.1.17 DateTimeDiff

No description.

Inputs:

Type and Name	Description
DateTime one	
DateTime two	

ReturnType: double

3.1.18 DateToEOM

Returns the last day of the month

Inputs:

Type and Name	Description
Date date	The date for which we are calculating the last day of the month

Outputs:

Type and Name	Description
Date eom	The last day of the month as output

3.1.19 DayCountFraction

Computes the day count fraction as a number of years between two dates using a day count convention.

Inputs:

Type and Name	Description
Date startDate	Start date.
Date endDate	End date.
DayCountConv dcc	Day count convention. See DayCountConv (page 18).

ReturnType: double

3.1.20 DayOfWeek

Returns the day of the week as a number from 0 to 6, 0=Sunday, 1=Monday etc.

Inputs:

Type and Name	Description
Date date	Date for which we are calculating the day of the week.

ReturnType: Weekday

3.1.21 DaysDiff

Computes the number of days difference between two dates using a day count convention.

Inputs:

Type and Name	Description
Date startDate	Start date.
Date endDate	End date.
DayCountConv dcc	Day count convention. See DayCountConv (page 18).

ReturnType: int

3.1.22 DaysInMonth

Returns the number of days in the month as a number from 1 to 31 using the Gregorian calendar.

We also require the year in case we are in a leap year and the month is February.

Inputs:

Type and Name	Description
int year	Year A.D. - typically 4 digits
int month	Month from 1 to 12. 1=January, 12=December etc

ReturnType: int

3.1.23 DiscountToRate

Converts a discount factor to a rate using day count convention and rate type.

Inputs:

Type and Name	Description
double discount	The discount factor
Date startDate	The start date for the requested rate
Date maturityDate	The maturity date for the requested rate
DayCountConv dayCount	The day count convention for the requested rate See DayCountConv (page 18).
RateType rateType	The type of the requested rate, e.g. simple, compound etc See RateType (page 23).

ReturnType: double

3.1.24 FileTime

No description.

Inputs:

Type and Name	Description
string filename	

ReturnType: DateTime

3.1.25 ForwardRate

No description.

Inputs:

Type and Name	Description
ZeroCurve zc	See ZeroCurve (page 24).
Date startDate	
Date maturityDate	
DayCountConv dcc	See DayCountConv (page 18).
RateType basis	See RateType (page 23).

ReturnType: double

3.1.26 HolidaysGetDates

No description.

Inputs:

Type and Name	Description
Holidays holidays	See Holidays (page 7).

ReturnType: Date[]

3.1.27 HolidaysLoadFile

This function loads a holiday file into memory. This is particularly useful if the holiday file has changed on the disk.

Inputs:

Type and Name	Description
Holidays fileName	See Holidays (page 7).

ReturnType: int

3.1.28 IsBusinessDay

No description.

Inputs:

Type and Name	Description
Date date	
Holidays holidays	See Holidays (page 7).

ReturnType: bool

3.1.29 IsHoliday

No description.

Inputs:

Type and Name	Description
Date date	
Holidays holidays	See Holidays (page 7).

ReturnType: bool

3.1.30 IsWeekend

No description.

Inputs:

Type and Name	Description
Date date	

ReturnType: bool

3.1.31 MMRate

No description.

Inputs:

Type and Name	Description
ZeroCurve zc	See ZeroCurve (page 24).
Date startDate	
Date maturityDate	
DayCountConv dcc	See DayCountConv (page 18).

ReturnType: double

3.1.32 MarketRate

No description.

Inputs:

Type and Name	Description
ZeroCurve zc	See ZeroCurve (page 24).
CashOrSwap cashOrSwap	See CashOrSwap (page 8).
Date startDate	
Date maturityDate	
DayCountConv cashDcc	See DayCountConv (page 18).
DateInterval swapFixedInterval	See DateInterval (page 13).
DayCountConv swapFixedDcc	See DayCountConv (page 18).
BadDayConv accBadDayConv	See BadDayConv (page 8).
BadDayConv payBadDayConv	See BadDayConv (page 8).
Holidays holidays	See Holidays (page 7).
bool stubAtEnd	Optional (default = `False`).

ReturnType: double

3.1.33 Math.Add

Adds upto three numbers together.

Inputs:

Type and Name	Description
double x	First number to be summed.
double y	Second number to be summed.
double z	Optional (default = `0`). Third number to be summed - if not provided then it will use 0.

ReturnType: double

3.1.34 Math.BinaryOp

Performs a binary operation on two numbers - return x op y.

Inputs:

Type and Name	Description

Continued on next page

Type and Name	Description
double x	The first number
Math.BinaryOperator op	The binary operator applied See Math.BinaryOperator (page 9).
double y	The second number

ReturnType: double

3.1.35 Math.LinearInterp

No description.

Inputs:

Type and Name	Description
int xs[]	
double fxs[]	
double x	

ReturnType: double

3.1.36 Now

No description.

Inputs: None

ReturnType: DateTime

3.1.37 RateToDiscount

Converts a rate to a discount factor using a day count conventions and rate type.

Inputs:

Type and Name	Description
double rate	The rate, e.g. 0.03 = 3%
Date startDate	The start date for the rate
Date maturityDate	The maturity date for the rate
DayCountConv dayCount	The day count convention for the rate See DayCountConv (page 18).
RateType rateType	The type of the rate, e.g. simple, compound etc See RateType (page 23).

ReturnType: double

3.1.38 StartOfMonths

No description.

Inputs:

Type and Name	Description
int year	

Outputs:

Type and Name	Description
Date startOfMonths[]	

3.1.39 StartOfYear

No description.

Inputs:

Type and Name	Description
int year	

Outputs:

Type and Name	Description
Date startOfYear	

3.1.40 SwapFixedFlows

This function calculates the cash flow dates and amounts associated with the fixed leg of an interest rate swap.

The calculations begin at the start date, with dates separated by the given date interval, and ends on the maturity date.

WARNING: This routine uses cash flow and yield conventions associated with swaps. The conventions are different from those associated with bonds.

About the calculations:

A cash flow at start date is only include if initial negative principal cash flow flag is 1.

If the total interval between maturity date and start date is not a multiple of the coupon interval, then a stub period is required. If the stub location is at the front, then dates are counted backwards from the maturity date. If stub location is at the back, then dates are counted forward from the start date.

If a stub is required, then the stub payment is calculated using the stub type via the function JpmcdsStubPayment. Both the stub location and stub type are defined by the stub method object.

Stub Location Flag: The date counting method used in this function may produce seemingly unexpected results. For example, if start date is 30th September and maturity date is 31st October of the same year, and the coupon interval is 1 month, then for a front stub you will not get a stub, but for a back stub you will get a stub between 30th October and 31st October. You can avoid this issue by using the flexible end of month adjust date interval type F instead of M.

Inputs:

Type and Name	Description
double couponRate	
Date startDate	Start date of the swap

Continued on next page

Type and Name	Description
DateInterval couponInterval	Time between payments See DateInterval (page 13).
Date maturityDate	Maturity date of the swap
DayCountConv dayCountConv	Day count convention used for calculating the amounts See DayCountConv (page 18).
bool frontStub	Optional (default = `True`). If there is a stub is it at the front (or back)?
bool shortStub	Optional (default = `True`). If there is a stub is it short (or long)?
bool subtractInitial	Optional (default = `False`). Should we subtract an initial payment of -1
bool keepStartDate	Optional (default = `False`). Should we keep the start date with zero payment if there is a stub, i.e. return a date before the startDate in the case of a stub.
bool addFinal	Optional (default = `False`). Should we add a final payment of +1
BadDayConv accBadDayConv	Accrual bad day convention See BadDayConv (page 8).
BadDayConv payBadDayConv	Payment bad day convention See BadDayConv (page 8).
string holidayFile	Name of holiday file for determining whether a day is a business day

ReturnType: CashFlowList (see page 11)

3.1.41 SwapRate

No description.

Inputs:

Type and Name	Description
ZeroCurve zc	See ZeroCurve (page 24).
Date startDate	
DateInterval couponInterval	See DateInterval (page 13).
Date maturityDate	
DayCountConv dcc	See DayCountConv (page 18).
BadDayConv accBadDayConv	See BadDayConv (page 8).
BadDayConv payBadDayConv	See BadDayConv (page 8).
Holidays holidays	See Holidays (page 7).
bool stubAtEnd	Optional (default = `False`).

ReturnType: double

3.1.42 Timestamp

Shows the timestamp of the build.

Inputs: None

ReturnType: string

3.1.43 UserGroups

No description.

Inputs:

Type and Name	Description
string serverName	Optional (default = `""`).
string userName	Optional (default = `""`).

ReturnType: string`[]`

3.1.44 UserName

No description.

Inputs: None

ReturnType: string

3.1.45 Version

Shows the version number as a string. You can request details of the build as well.

Inputs:

Type and Name	Description
bool showDetails	Optional (default = `True`). Show build details as well as just the version number

ReturnType: string

3.1.46 YMD

No description.

Inputs:

Type and Name	Description
Date date	

Outputs:

Type and Name	Description
int year	
int month	
int day	

3.1.47 Year

No description.

Inputs:

Type and Name	Description
Date date	

Outputs:

Type and Name	Description
int year	

3.1.48 ZeroCurveCash

No description.

Inputs:

Type and Name	Description
Date baseDate	
Date dates[]	
double rates[]	
DayCountConv dayCountConv	See DayCountConv (page 18).
InterpType interpType	See InterpType (page 8).

ReturnType: ZeroCurve (see page 24)

3.1.49 ZeroCurveCashSwaps

No description.

Inputs:

Type and Name	Description
Date baseDate	
CashOrSwap cashOrSwaps[]	Optional (default = ""). See CashOrSwap (page 8).
Date dates[]	Optional.
double rates[]	Optional (default = 0).
DayCountConv cashDcc	See DayCountConv (page 18).
InterpType interpType	See InterpType (page 8).
DateInterval swapFixedInterval	See DateInterval (page 13).
DayCountConv swapFixedDcc	See DayCountConv (page 18).
BadDayConv badDayConv	See BadDayConv (page 8).
Holidays holidays	See Holidays (page 7).
string flags	Optional (default = "").

ReturnType: ZeroCurve (see page 24)

3.1.50 ZeroCurveSample

Generates a sample zero curve out to 10 years with slope and some extra shape which goes up and decays to zero.

Not to be used for pricing - simply a device to get example curves with a bit of shape.

Inputs:

Type and Name	Description
Date baseDate	
double baseRate	

Continued on next page

Type and Name	Description
double slope	
double hump	Optional (default = `0`).
InterpType interpType	See InterpType (page 8).

ReturnType: ZeroCurve (see page 24)

3.1.51 ZeroCurveSwaps

No description.

Inputs:

Type and Name	Description
ZeroCurve stubCurve	See ZeroCurve (page 24).
Date swapDates`[]`	
double swapRates`[]`	
DateInterval couponInterval	See DateInterval (page 13).
DayCountConv fixedDcc	See DayCountConv (page 18).
BadDayConv badDayConv	See BadDayConv (page 8).
Holidays holidays	See Holidays (page 7).

ReturnType: ZeroCurve (see page 24)

3.2 Class methods

These are the class methods (including `static` methods in the library).

Note that when using the class method within Excel you will need to use the overall namespace for the library as a prefix to the full class method name.

From within Python and .NET then you only need the last part of the full class method name (using `'.'` as the separator and replace the rest of the full class method name with an instance of that class (which is the first part of the full class method name).

There is an exception for `static` methods. In this case the calling convention is the same for Python and .NET as for Excel - i.e. you need the full class name prefixed by the overall namespace for the library.

3.2.1 CashSwapCurve.CashOrSwaps

No description.

Inputs: None

ReturnType: CashOrSwap[]

3.2.2 CashSwapCurve.ZeroCurve

No description.

Inputs:

Type and Name	Description
InterpType interpType	See InterpType (page 8).
string flags	Optional (default = "").

ReturnType: ZeroCurve (see page 24)

3.2.3 DateInterval.Make

Constructs DateInterval from number of periods and period type.

Inputs:

Type and Name	Description
int numPeriods	Number of periods.
char periodType	Period type, e.g. M,D,W,Y etc.

ReturnType: DateInterval (see page 13)

3.2.4 Math.Matrix.Cell

Returns an individual cell of the matrix.

Inputs:

Type and Name	Description
int i	
int j	

ReturnType: double

3.2.5 Math.Matrix.Cols

Returns the number of columns of the matrix.

Inputs: None

ReturnType: int

3.2.6 Math.Matrix.Data

Returns all the data for the matrix.

Should give the same result as accessing the data attribute.

Inputs: None

ReturnType: double[,]

3.2.7 Math.Matrix.Rows

Returns the number of rows of the matrix.

Inputs: None

ReturnType: int

3.2.8 Math.Matrix.Size

No description.

Inputs: None

Outputs:

Type and Name	Description
int rows	
int cols	

3.2.9 RateType.Make

No description.

Inputs:

Type and Name	Description

Continued on next page

Type and Name	Description
string name	

ReturnType: RateType (see page 23)

3.2.10 ZeroCurve.FV

No description.

Inputs:

Type and Name	Description
Date startDate	
Date endDate	

ReturnType: double

3.2.11 ZeroCurve.PV

No description.

Inputs:

Type and Name	Description
Date interpDate	

ReturnType: double

3.2.12 ZeroCurve.ZeroRate

No description.

Inputs:

Type and Name	Description
Date interpDate	

ReturnType: double