4 Dates: Revised

In the Fundamentals Tutorial chapter, we have introduced the roll schedule, the market convention, and the start date. In this chapter, we will follow these concepts and discuss the following in detail:

- Concepts of rolls and roll schedules
- Concepts of a market convention and its constituent components
- Replicate a built-in market convention
- Construct a custom market convention
- Concepts of the trade date and start date

4.1 Rolls and Rolls Schedules

A roll and a roll schedule, is an important concept which is used frequently in F3. You have seen a roll schedule in the Fundamentals Tutorial chapter, where it was used in constructing a fixed leg, a floating leg, and an interest rate swap. We now discuss the concepts of the rolls and the roll schedules.

4.1.1 What Are Rolls and Rolls Schedules

Concept of A Roll

A roll encapsulates the concept of a single payment in a series of payments.

- **Case I**: At the minimum, a roll consists of a payment date and an accrual fraction. In this case, the accrual fraction is set to 1.0

  > The accrual fraction is often calculated by the function DayCountFraction. It is also called the accrual fraction of the roll.

- **Case II**: More frequently, a roll is described by the following pieces of information, and has this format:

Payment date	Accrual fraction of the roll	Start date	End date

- In particular, the start date and end date information is needed where the payment is a function of a time period determined by the two dates. Although the accrual fraction of the roll is often calculated by the DayCountFraction function, it is not calculated directly from the start date and the end date of the roll.
- Examples of a roll are as follows:
  - A date specifying when a payment is made, followed by the associated accrual fraction, such as:

2014-12-18	0.5

  - A date specifying when a payment is made, followed by the associated accrual fraction, start date and end date, such as:

2014-12-18	0.508333	2014-06-18	2014-12-18

Concept of A Roll Schedule

- **Case I**: A roll schedule is a collection of rolls, and has the following format:

Payment date	Accrual fraction of the roll	Start date	End date
Payment date	Accrual fraction of the roll	Start date	End date
Payment date	Accrual fraction of the roll	Start date	End date
...	...	...	...
Payment date	Accrual fraction of the roll	Start date	End date

- **Case II**: Alternatively, a roll schedule can be represented as a three-column array, and has the following format:

S	M	C

- where
  - **S** is the start date of the roll schedule
  - **M** is a maturity descriptor giving the end point of the schedule
  - **C** is the market convention object that generates the schedule
- Examples of a roll schedule are as follows:
  - The following table is an example of a roll schedule:

2014-12-04	0.252777778	2014-09-04	2014-12-04
2015-03-04	0.25	2014-12-04	2015-03-04
2015-06-04	0.255555556	2015-03-04	2015-06-04
2015-09-04	0.255555556	2015-06-04	2015-09-04

- Note that every row of this table is also a roll. The aggregate of the rolls is the roll schedule.
- The following three-column array is an example of the alternative representation a roll schedule:

2014-09-25	5y	SwapUSD3m

4.1.2 Generate Roll Schedules: Two Ways

4.1.2.1 Use RollSchedule Function

The RollSchedule function constructs a schedule of rolls, which starts on the specified start date, ends on the end date, is calculated according to the specified market convention, and has the length specified by the maturity descriptor. This is illustrated in Fig. 4.1.

RollSchedule				
StartDate	2015-09-25			
Maturity	6m			
MarketConvention	SwapUSD3m			
	2015-12-28	**0.261111111**	**2015-09-25**	**2015-12-28**
	2016-03-25	**0.244444444**	**2015-12-28**	**2016-03-25**

Fig. 4.1. Call to RollSchedule

The arguments of RollSchedule are:

1. *StartDate*: The start date.
2. *Maturity*: The description of maturity. Here we have used **6m** for six months from the start date.
3. *MarketConventions*: The market convention to use for the schedule, which is set to **SwapUSD3m**.

> RollSchedule returns a four-column array. You need to highlight multiple rows of the four-column array containing the function call and press F2. Then press CRTL + SHIFT + ENTER. Note that the exact number of rows you need depends on the combination of the maturity descriptor and the market conventions used.

> You need to re-format the returned first, third and fourth column as **short dates** to display them as dates.

The generated roll schedule can then be used as an input wherever the *RollSchedule* argument is required.

4.1.2.2 Use Schedule Generators

Whenever an input for the *RollSchedule* argument is required, you can also pass the inputs of RollSchedule function directly into the field. To do this, the input should be given in a three-column array of **StartDate**, **Maturity**, and **MarketConvention**. This is demonstrated in Fig. 4.2, where we created a fixed leg using this method.

CreateSingleCurrencyFixedLeg			
ProductName	FixedLeg		
RollSchedule	2015-09-25	6m	SwapUSD3m
FixedCoupon	2		
Notionals	10 mio		
Currency	USD		
PayRec	Pay		
Index	UnitConstant		
	FixedLeg		

Fig. 4.2. Call to CreateSingleCurrencyFixedLeg

> Using RollSchedule and Using schedule generators are essentially the same method. In Subsubsec. 4.1.2.1,

you used RollSchedule to generate the schedule, which can then be linked to the input cell in the CreateSingleCurrencyFixedLeg. In Subsubsec. 4.1.2.1, the generator also calls the RollSchedule when you supply a three-column array of a start date, a maturity descriptor, and a market convention.

4.2 Market Conventions

4.2.1 What Are Market Conventions, and What Do They Do

A market convention is a date-related convention of a market that governs several date-related behaviours of the market. It is a higher-level convention that collects the definitions of the following lower-level, market-specific, date-related conventions in one central location:

- Holiday convention
- Day count convention
- Settlement delay

Additionally, a market convention also holds the prescriptions for the following:

- Calculating and adjusting dates
- Generating roll schedules that are not of equal length

A market convention is a composite of the above lower-level conventions and prescription. Together, they form a market convention.

You can use a market convention object to do the following:

- To form a roll schedule
- To generate a start date from trade date, or to bound a trade date from a start date
- To determine if a date is a valid business day
- To calculate a maturity date

4.2.2 Major Types of Market Conventions

Market convention objects are stored in the MarketConventions repository. Although there is a large number of market convention objects in this repository, there are three major types:

Convention Type	Example	Format	Remark
Liquid swap markets	SwapUSD3m SwapUSDAnnual	"Swap" + "Currency Name" + "Tenor"	If the tenor description ends with Semi or Annual, then the convention is intended for use in the fixed leg of a swap. Therefore, every such market convention, such as SwapUSDAnnual, is synonymous with the corresponding convention ending with "Fixed". If the tenor description ends with the number of month 3, 6, or 12, then the convention is intended for use in the floating leg of a swap. For this reason, every such market convention, such as SwapUSD3m, is synonymous with the corresponding convention ending with "Floating".
Cash deposits	CashUSD	"Cash" + "Currency Name"	
Daily schedules	NewYorkDaily	"Holiday convention name" + "Daily"	

4.2.3 What Goes Into a Market Convention: SwapUSD3m as an Example

4.2.3.1 Payment Holidays: Holiday Conventions

A market convention must hold the definition of business days for a market. The job of defining a business day is accomplished by a holiday convention.

A holiday convention determines whether a particular date is a valid business day in a given financial market. Thus, different jurisdictions and exchanges have different holiday conventions. Numerous holiday conventions are represented as objects, and are stored in the HolidayConvention repository.

Some examples of holiday convention objects include:

- London: London holidays, plus Saturdays and Sundays
- NewYork: New York holidays, plus Saturdays and Sundays
- Brasilia Fed Dist: Brasilia Fed Dist holidays, plus Saturdays and Sundays

4.2.3.1.1 Construct Holiday Conventions From a List of Dates

You can create any non-standard holiday convention using the CreateHolidayConventionFromDates function. CreateHolidayConventionFromDates takes a list of dates that you supply to form a new holiday convention. This is illustrated in Fig. 4.3

CreateHolidayConventionFromDates				
HolidayConventionName	Holiday_Dates			
ListOfDates	2015-06-24	2015-05-23	2015-12-27	2015-04-08
	Holiday_Dates			

Fig. 4.3. Call to CreateHolidayConventionFromDates

The arguments of CreateHolidayConventionFromDates are:

- *HolidayConventionName*: Name to use for the new holiday convention. We use **Holiday_Dates**.
- *ListOfDates*: List of dates that you supply to build the new holiday convention. You can specify as many dates as you like.

> In the function call, press F2. Change the formula of the function call such that it captures the entire array of dates in the function arguments, and press CTRL + SHIFT + ENTER.

4.2.3.1.2 Construct Holiday Conventions From a List of Conventions

CreateCompositeHolidayConvention takes a list of constituent holiday conventions to make a new, composite holiday convention. See Fig. 4.17.

CreateCompositeHolidayConvention		
HolidayConventionName	Custom_Holiday	
HolidayConventions	Holiday_Dates	Weekends
	Custom_Holiday	

Fig. 4.17. Call to CreateCompositeHolidayConvention

The arguments of CreateCompositeHolidayConvention are:

- *HolidayConventionName*: Name to use for the new holiday convention. We use **Custom_Holiday**.
- *HolidayConventions*: List of holiday conventions you supply, which can either be holiday conventions that you constructed or built-in holiday conventions in the repository. Here, we use **Holiday_Dates** and **Weekends** to build the new holiday convention.

Custom_Holiday you just constructed is a holiday convention that defines all weekends and the list of dates you specified as holidays.

4.2.3.1.3 Determine if a Date is a Valid Business Day

You use the function IsGoodBusinessDay to determine if a date is a valid business day given a holiday convention. For example, in Fig. 4.5, we check whether July 14, 2014 is a valid business day in the New York holiday convention.

IsGoodBusinessDay	
Date	2015-07-14
HolidayConventions	Custom_Holiday
	TRUE

Fig. 4.5. Call to IsGoodBusinessDay

The arguments of IsGoodBusinessDay are:

- *Date*: Date that you want to verify
- *HolidayConvention*: Holiday convention to use. Here we used the holiday convention you just constructed, **Custom_Holiday**.

IsGoodBusinessDay returns either True or False. True means that the date in question is a valid business day for the New York market.

4.2.3.1.4 Find out When the Next Holiday Is

To find out the holidays under a particular market convention, use ListExplicitHolidayDates. See Fig. 4.6.

ListExplicitHolidayDates	
StartDate	2014-10-01
EndDate	2014-10-31
Conventions	Custom_Holiday
	2014-10-04
	2014-10-05
	2014-10-11
	2014-10-12
	2014-10-18
	2014-10-19
	2014-10-25
	2014-10-26

Fig. 4.6. Call to ListExplicitHolidayDates

The arguments of ListExplicitHolidayDates are:

- *StartDate*: First date in the range, **October 1, 2014**.
- *EndDate*: Last day in the range, **October 31, 2014**.
- *conventions*: Convention or list of conventions used to retrieve the holidays. We use **Custom_Holiday** you constructed above.

The ListExplicitHolidayDates returns a multi-row column of dates, each of which is a holiday under the specified convention. To display the full result of the function output, highlight sufficient number of cells (vertically), and press CTRL + SHIFT + ENTER. Re-format the output as short dates.

4.2.3.2 Day Count Convention

A market convention must hold the prescription of how to calculate the length of time between two dates. The job of prescribing the method to determine this period of time is accomplished by a day count convention.

For example, a day count convention is required to determine the amount of interest that accrues over a period of time, or the length of time in discounting of future cash flows to their present values.

Day count conventions are represented as objects, and are stored in the DayCountConvention repository.

4.2.3.3 Settlement Delay

A market convention must also contain a definition of a settlement delay, or a spot lag. In practice, the start (or the settlement date) of a trade is usually a certain lag after the trade date. The most used lag is two business days, however, other spot lags are possible.

The existence of a spot lag is due to the settlement issues when the counterparties are located in different time zones. When this happens, the agreeing to and the confirming of trades are more time-consuming and complex, and the settlement delay (lag) allows for a mutually agreeable time for the settlement to complete.

A valid input for a settlement delay must be one of the following maturity descriptors:

- A date
- A short rate descriptor, such as:
  - o/n, on, O/N, or ON: overnight
  - t/n, tn, T/N, or TN: tomorrow next
  - s/n, sn, S/N, or SN: spot next
  - s/w, sw, S/W, or SW: spot week

  For example, let the trade date be Thursday March 5, and the spot date be 2b forward, which is Monday March 9. The o/n rate is the 1b rate starting on the trade date: Thursday March 5 to Friday March 6. The t/n rate is the 1b rate starting 1b after the trade date: Friday March 6 to Monday March 9. The s/n rate is the 1b rate out of spot: Monday March 9 to Tuesday March 10. And the s/w rate is the one week rate out of spot: Monday March 9 to Monday March 16.
- A number of days, business days, months and years. For example:
  - $nb$is $n$business days.
  - $nd$is $n$calendar days.
  - $nw$is $n$weeks.
  - $nm$is $n$months.
  - $ny$is $n$years.

  You can concatenate the above expressions for your purposes. For example, "3m2d" or "1y 6m" (note that the spaces are optional).

- More input types are available. For more details of valid inputs of a maturity descriptor, see Maturity Descriptor in F3 Framework Reference Manual.

4.2.3.4 Calculating and Adjusting Dates

In general, a market convention must contain the prescriptions to:

- Calculate dates
- Adjust dates

We use the **SwapUSD3m** market convention as an example. The dates are computed from the start date. The last date will be the start date plus the total length of the trade. The intermediary dates are spaced by the given payment period (roll length) except potentially one. The non-standard period is call the stub, is the first one in the case of SwapUSD3m (see Stub Type). All the dates will be computed first without any adjustment. Then they are adjusted all at once by the month-end rule and the date modifiers, which is discussed below.

In F3, the following three objects together provide prescriptions of calculating and adjusting dates:

- Maturity calculators
- Settlement maturity calculator
- Roll day modifier

4.2.3.4.1 Maturity Calculators

In F3, the maturity date is calculated by a simple or compound maturity calculator. The construction of a maturity calculator is illustrated in the following diagram:

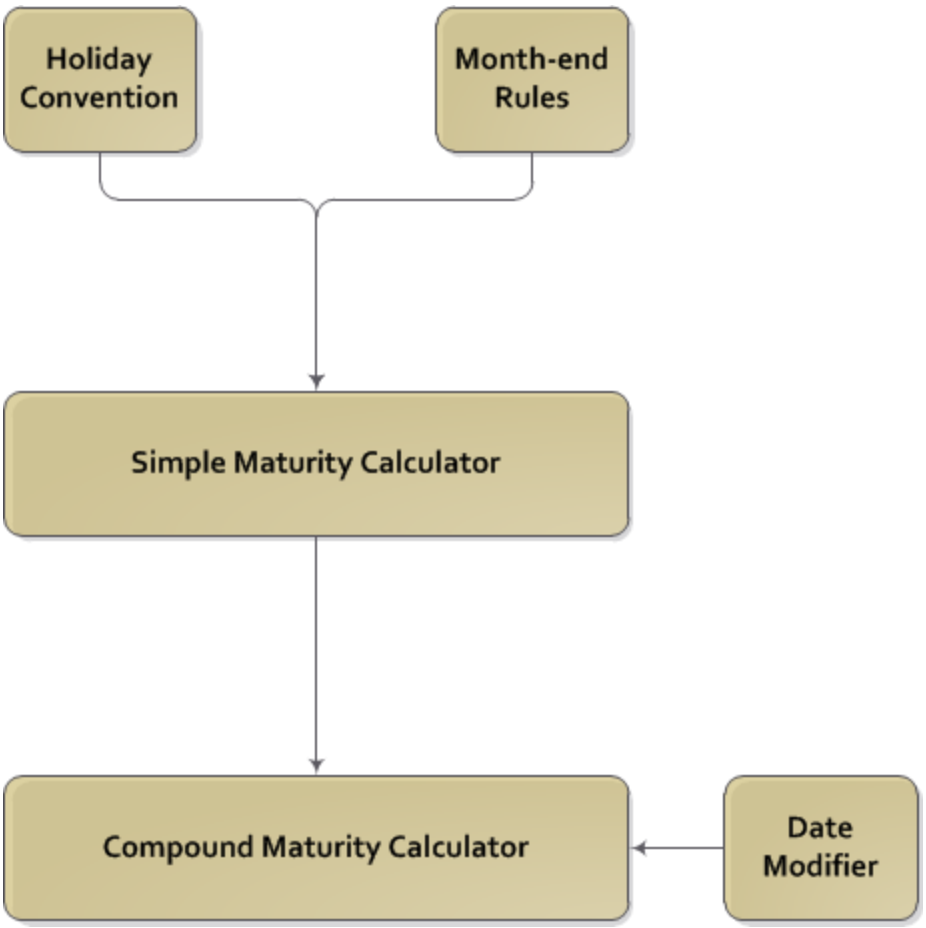

Fig. 4.7. Anatomy of a maturity calculator

As shown in the Fig. 4.7, a simple maturity calculator incorporates both the holiday convention and the month-end rules. A compound maturity calculator is then built upon the simple maturity calculator and a date modifier. Therefore, this compound maturity calculator not only takes into account the holiday convention of a given market, it also takes into account the month-end rules. Furthermore, in case the calculated date happens to fall on a weekend, it also automatically adjusts the calculated maturity date to a good business day.

The month-end rule determines the treatment of a specific date if the start date falls on the end of the month:

- PreserveMonthEnd:
- MoveIntoNextMonth:

Your choice of date modifier will also impact how the calculated date is adjusted. For example, given the holiday convention NewYork, NewYorkFoll adjusts the date to the nearest future business day, whereas NewYorkModFoll adjusts the date to the nearest business day, unless this change moves the date out of the month, in which case, the date is moved to the first preceding business day.

In Fig. 4.8, we show an example where we determine the maturity date using a compound maturity calculator:

MaturityDate	
StartDate	2015-01-02
Maturity	1y
MaturityConvention	NewYorkModFoll
	04-Jan-16

Fig. 4.8. Call to MaturityDate

The arguments for MaturityDate are:

- *StartDate*: **2015-01-02**.
- *Maturity*: input must be of a Maturity descriptor type, such as **1y**.
- *MaturityConvention*: input is either an element in the MaturityCalculator or MarketConventions repository. In this example, we supplied a custom maturity calculator, **NewYorkModFoll**.

4.2.3.4.2 Date Modifiers

The DateModifier is a convention for adjusting dates when a specified date is not a good business day. The following is a brief explanation of some notations used in the date modifiers:

- Foll: Following business day. If a date falls on a weekend or a holiday, the next good business day is used.
- ModFoll: Modified following business day. If a date falls on a weekend or a holiday, the next good business day is used. If the next good business day falls in the next month, the previous good business day is used.
- Prec: Preceding business day. If a date falls on a weekend or a holiday, the preceding good business day is used.
- ModPrec: Modified preceding business day. If a date falls on a weekend or a holiday, the previous good business day is used. If the previous good business day falls in the previous month, the following good business day is used.

4.2.3.5 Roll Length

For a swap market, a market convention must also include the definition of the length of time between payments. For example, a fixed leg of a swap paying 2% per year on a quarterly basis will have a roll length (or roll tenor) of 3 months.

4.2.3.6 Stub Type

For a swap market, a market convention must provide instructions on how to generate roll schedules that are not of equal length.

The start of the swap is usually a certain lag (settlement delay) after the trade date. The payments on the fixed leg are regularly spaced by a given period (most of them with a 6-month or 12-month period). Similarly, the payments on the floating leg are also regularly spaced (most of them with 3-month or 6-month period). The roll of irregular length is called the stub.

The stub type specifies where to place the stub, if any, and whether the stub should be longer or shorter than a regular roll. See Argument type: stub type for more information.

4.2.4 Construct a Market Convention Equivalent to SwapUSD3m

In this section, we examine the built-in swap market convention **SwapUSD3m** closely. We show that we can replicate it by building it from scratch, using the following F3 functions:

- CreateSimpleMaturityCalculator
- CreateModifyingMaturityCalculator
- CreateSettlementMarketConvention
- CreateMonthlyScheduleMarketConvention

The conceptual structure of SwapUSD3m is as follows:

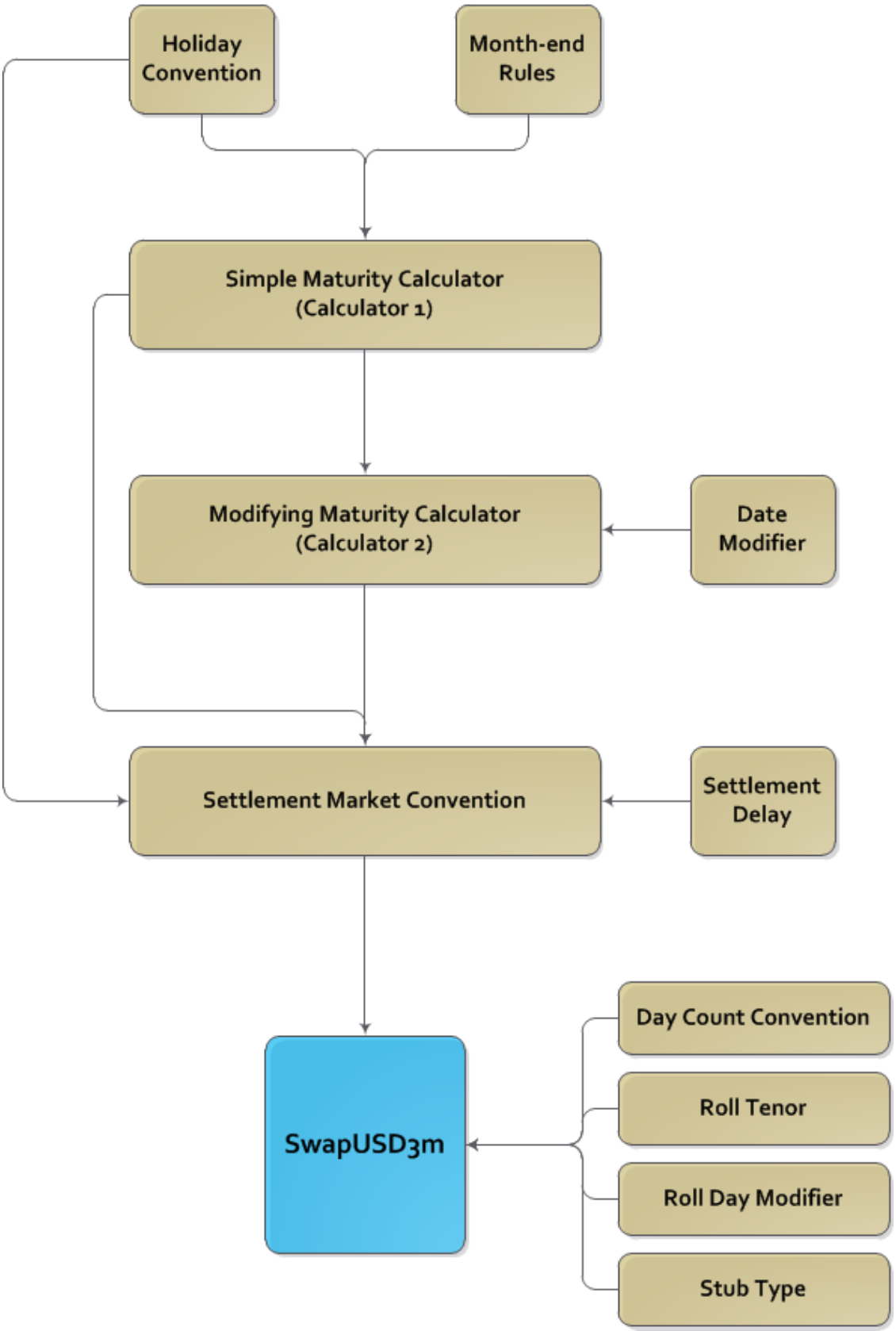

Fig. 4.9. Conceptual Structure of SwapUSD3m

To replicate SwapUSD3m, perform the following steps:

1. Call the function CreateSimpleMaturityCalculator, which is illustrated in Fig. 4.10.

CreateSimpleMaturityCalculator	
MaturityCalculatorName	Calculator1
HolidayConventions	NewYork
PreserveMonthEnd	FALSE
MoveIntoNextMonth	FALSE
	Calculator1

Fig. 4.10. Call to CreateSimpleMaturityCalculator (Calculator1)

   The arguments of CreateSimpleMaturityCalculator are:

   - *MaturityCalculatorName*: **Calculator1**.
   - *HolidayConvention*: Holiday convention for payment dates, **NewYork**
   - *PreserveMonthEnd*: Optional. Flag to ensure month-end dates are preserved under month changes. Set to **FALSE**.
   - *MoveIntoNextMonth*: Optional. Flag to allow movement into a following month. Set to **FALSE**.

2. Call the function CreateModifyingMaturityCalculator. This is illustrated in Fig. 4.11.

CreateModifyingMaturityCalculator	
MaturityCalculatorName	Calculator2
UnderlyingMaturityCalculator	Calculator1
DateModifier	NewYorkModFoll
	Calculator2

Fig. 4.11. Call to CreateSimpleMaturityCalculator (Calculator2)

   The arguments of CreateModifyingMaturityCalculator are:

   - *MaturityCalculatorName*: **Calculator2**
   - *UnderlyingMaturityCalculator*: **Calculator1** is used as the underlying maturity calculator

- *DateModifier*: Optional. Set to **NewYorkModFoll**.

3. Call the function CreateSettlementMarketConvention. This is illustrated in Fig. 4.12.

CreateSettlementMarketConvention	
MaturityConventionName	Convention
PaymentHolidays	NewYork
MaturityCalculator	Calculator1
SettlementMaturityDescriptor	2b
SettlementMaturityCalculator	Calculator2
	Convention

Fig. 4.12. Call to CreateSettlementMarketConvention (Convention)

The arguments of CreateSettlementMarketConvention are:

- *MarketConventionName*: **Convention**
- *PaymentHolidays*: Holiday convention for payment dates, **NewYork**
- *MaturityCalculator*: Maturity calculator for the rolls, **Calculator1**
- *SettlementMaturityDescriptor*: Settlement delay, **2b**
- *SettlementMaturityCalculator*: Calculator for the settlement date, **Calculator2**

4. Call the function CreateMonthlyScheduleMarketConvention. This is illustrated in Fig. 4.13.

CreateMonthlyScheduleMarketConvention	
MaturityConventionName	Replicated
UnderlyingConventions	Convention
RollLength	3
StubType	StubAtStart
RollDayModifier	NewYorkModFoll
DayCountConvention	act/360
ModifyingStart	FALSE
EndDateModifier	NewYorkModFoll
MonthEndMethod	NoAdjustment
	Replicated

Fig. 4.13. Call to CreateMonthlyScheduleMarketConvention (Replicated)

The arguments of CreateMonthlyScheduleMarketConvention are:

- *MarketConventionName*: **Replicated**
- *UnderlyingConventions*: Underlying convention(s) to use, **Convention**
- *RollLength*: Length of roll, in months. The input must be in an integer. We use **3**
- *StubType*: **StubAtStart**
- *RollDayModifier*: Date modifier that is applied to all roll end dates, except for the final roll. We use **NewYorkModFoll**.
- *DayCountConvention*: **act/360**.
- *ModifyStart*: Optional. Default: **FALSE**. Flag to indicate if the start date should also be adjusted using the *RollDayModifier*.
- *EndDateModifier*: Optional. Default: **NullModifier**. This input specifies the date modifier for the end date of the final roll.
- *MonthEndMethod*: Optional. Default: **NoAdjustment**. This input specifies how to generate a schedule when the schedule end date falls on the month end.

You can generate two roll schedules using the function RollSchedule: one uses SwapUSD3m, the other one uses the constructed market convention Replicated. You can compare that the roll schedules generated under these two conventions are, in indeed, identical. This is illustrated in Fig. 4.14 and Fig. 4.15, respectively.

RollSchedule				
StartDate	2014-01-02			
Maturity	1y			
MarketConvention	SwapUSD3m			
	2015-04-02	0.250000000	2015-01-02	2015-04-02
	2015-07-02	0.252777778	2015-04-02	2015-07-02
	2015-10-02	0.255555556	2015-07-02	2015-10-02
	2016-01-16	0.261111111	2015-10-02	2016-01-04

Fig. 4.14. Call to RollSchedule using SwapUSD3m as Market Convention

RollSchedule				
StartDate	2014-01-02			
Maturity	1y			
MarketConvention	Replicated			
	2015-04-02	0.250000000	2015-01-02	2015-04-02
	2015-07-02	0.252777778	2015-04-02	2015-07-02
	2015-10-02	0.255555556	2015-07-02	2015-10-02
	2016-01-16	0.261111111	2015-10-02	2016-01-04

Fig. 4.15. Call to RollSchedule using Replicated as Market Convention

4.2.5 Construct a Market Convention For a Non-Standard Liquid SwapMarket

In this section, we construct a custom market convention that incorporates the valid business days in both Toronto and Moscow, but with "weekends" now redefined as all Sundays and Mondays. Additionally, we also require that the payment dates are adjusted to be after the 20th day of the month. This custom market convention uses the day count convention of 30E+/360.

The conceptual structure of this custom market convention is as follows:

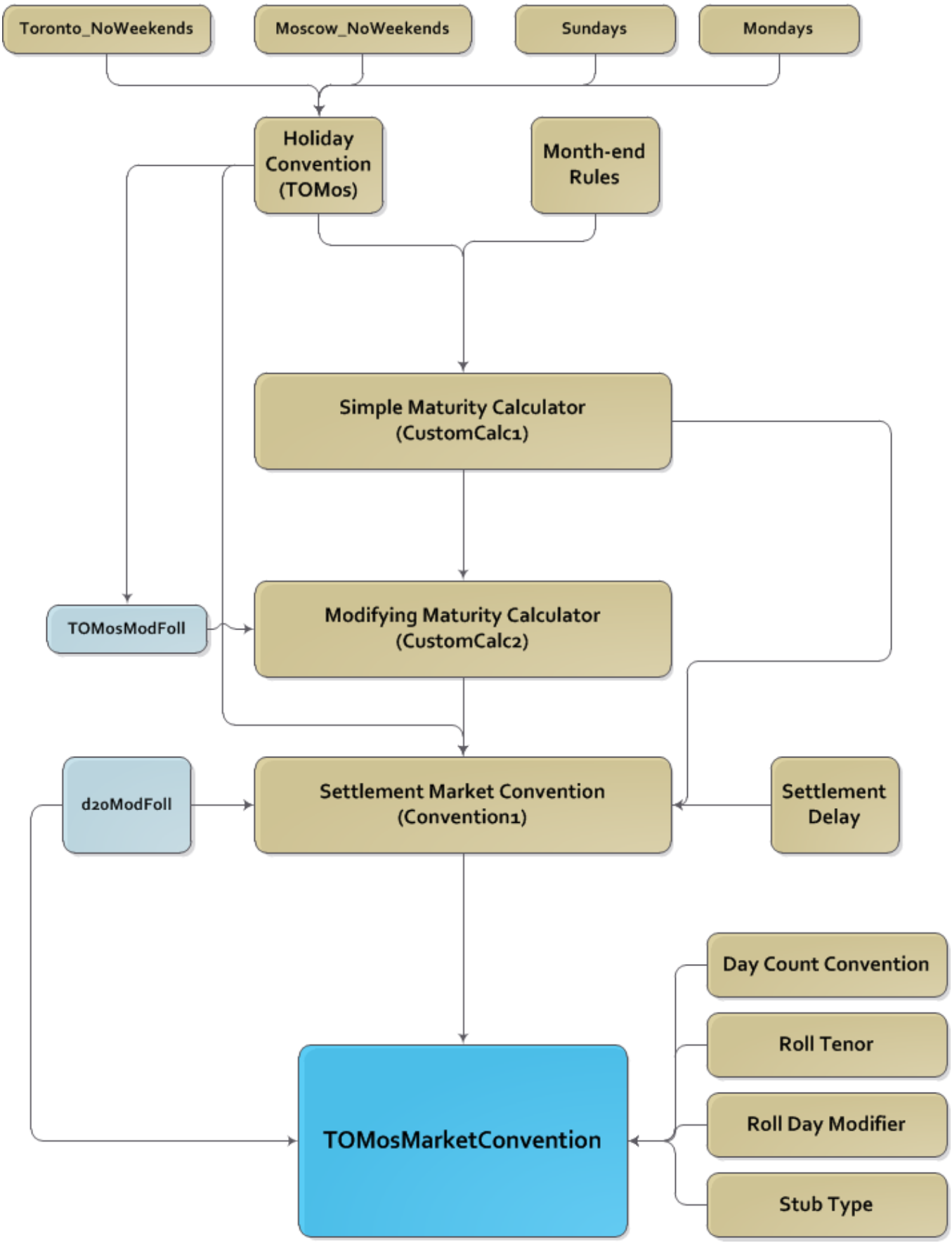
Toronto_NoWeekends
Moscow_NoWeekends
Sundays
Mondays
Holiday Convention (TOMos)
Month-end Rules
Simple Maturity Calculator (CustomCalc1)
TOMosModFoll
Modifying Maturity Calculator (CustomCalc2)
d2oModFoll
Settlement Market Convention (Convention1)
Settlement Delay
Day Count Convention
Roll Tenor
Roll Day Modifier
Stub Type
TOMosMarketConvention

Fig. 4.16. Conceptual Structure of TOMosMarketConvention

To construct TOMosMarketConvention, perform the following steps:

1. Call the function CreateCompositeHolidayConvention, which is illustrate in Fig. 4.17.

CreateCompositeHolidayConvention				
HolidayConventionName	TOMos			
HolidayConventions	Toronto_NoWeekends	Moscow_NoWeekends	Sundays	Mondays
	TOMos			

Fig. 4.17. Call to CreateCompositeHolidayConvention

2. Call the function CreateSimpleMaturityCalculator. This is illustrated in Fig. 4.18.

CreateSimpleMaturityCalculator	
MaturityCalculatorName	CustomCalc1
HolidayConventions	TOMos
PreserveMonthEnd	FALSE
MoveIntoNextMonth	FALSE
	CustomCalc1

Fig. 4.18. Call to CreateSimpleMaturityCalculator (CustomCalc1)

3. Call the function CreateFollowingDateModifier to create d20ModFoll modifier. This date modifier adjusts the payment dates to be the 20th day of the month. If the 20th day is not a business day according to the holiday convention TOMos, then the modifier adjusts the date to be the following good business day, unless the day is in the next calendar month, in which case, the adjusted date is the preceding good business day.

   The function call is illustrated in Fig. 4.19.

CreateFollowingDateModifier	
DateModifierName	d20ModFoll
UnderlyingDateModifier	d20
HolidayConventions	TOMos
AllowMonthChange	FALSE
	d20ModFoll

Fig. 4.19. Call to CreateFollowingDateModifier (d20ModFoll)

4. Call the function CreateFollowingDateModifier to create TOMosModFoll modifier. This date modifier adjusts the maturity dates to be the following business day if it is not a good business day according to the holiday convention TOMOs, unless the day is in the next calendar month, in which case, the adjusted date is the preceding good business day.

   The function call is illustrated in Fig. 4.20.

CreateFollowingDateModifier	
DateModifierName	TOMosModFoll
UnderlyingDateModifier	NullModifier
HolidayConventions	TOMos
AllowMonthChange	FALSE
	TOMosModFoll

Fig. 4.20. Call to CreateFollowingDateModifier (TOMosModFoll)

5. Call the function CreateModifyingMaturityCalculator. This is illustrated in Fig. 4.21.

CreateModifyingMaturityCalculator	
MaturityCalculatorName	CustomCalc2
UnderlyingMaturityCalculator	CustomCalc1
DateModifier	TOMosModFoll
	CustomCalc2

Fig. 4.21. Call to CreateModifyingMaturityCalculator (CustomCalc2)

6. Call the function CreateSettlementMarketConvention. This is illustrate d in Fig. 4.22.

CreateSettlementMarketConvention	
MarketConventionName	Convention1
PaymentHolidays	TOMos
MaturityCalculator	CustomCalc1
SettlementMaturityDescriptor	2b
SettlementMaturityCalculator	CustomCalc2
	Convention1

Fig. 4.22. Call to CreateSettlementMarketConvention (Convention1)

7. Call the function CreateMonthlyScheduleMarketConvention. This is illustrated in Fig. 4.23.

CreateMonthlyScheduleMarketConvention	
MarketConventionName	TOMosMarketConvention
UnderlyingConventions	Convention1
RollLength	3
StubType	StubAtStart
RollDayModifier	d20ModFoll
DayCountConvention	30E+/360
ModifyingStart	FALSE
EndDateModifier	NullModifier
MonthEndMethod	NoAdjustment
	TOMosMarketConvention

Fig. 4.23. Call to CreateMonthlyScheduleMarketConvention (TOMosMarketConvention)

4.3 Start Date

4.3.1 Determine a Start Date from a Trade Date

In practice, the start date is typically different from the trade date. The TradeDateToStartDate function determines the start date given a trade date, which is illustrate in Fig. 4.24. This start date is, in turn, used in all subsequent trade-related calculations. For example, the start date is used as an input to the function MaturityDate and RollSchedule.

TradeDateToStartDate	
TradeDate	2015-01-05
MarketConvention	SwapUSD3m
	2015-01-07

Fig. 4.24. Call to TradeDateToStartDate

4.3.2 Find the Trade Date From the Start Date

Sometimes you may need to determine a trade date from a start date. In this case, you can use the function TradeDatesBoundingStartDate, which is related to TradeDateToStartDate. This is illustrated in Fig. 4.25. TradeDatesBoundingStartDate gives a date range that bounds the start date, according to a specific market convention.

TradeDatesBoundingStartDate		
StartDate	2015-01-07	
MarketConvention	SwapUSD3m	
	2015-01-02	**2015-01-06**

Fig. 4.25. Call to TradeDatesBoundingStartDate

The first date returned in the array is the latest date that, when used together with a given market convention, will give a start date before the specified *StartDate*. Therefore, the first date provides a lower bound for the trade date.

The second date returned is the earliest date that, when used together with a given market convention, will give a start date after the specified *StartDate*. Therefore, it provides a upper bound for the trade date.

This means that the date(s) in between the lower and upper bounds, when used as a trade date, will give a start date that is equal to the specified date or, or an invalid business day. Therefore, when interpreted together, the actual *TradeDate* must lie between the lower bound and the upper bounds. In this example, this gives the actual start date of July 2, 2014.

> Note that if the lower bound is equal to the upper bound, then there is no valid trade date for the supplied *StartDate*.

<< PREVIOUS CHAPTER CONTENTS NEXT CHAPTER >>