reposit Version 1.8

and

The Future of Spreadsheet Addins

Eric Ehlers
Reposit Ltd
12 July 2016

Table of Contents

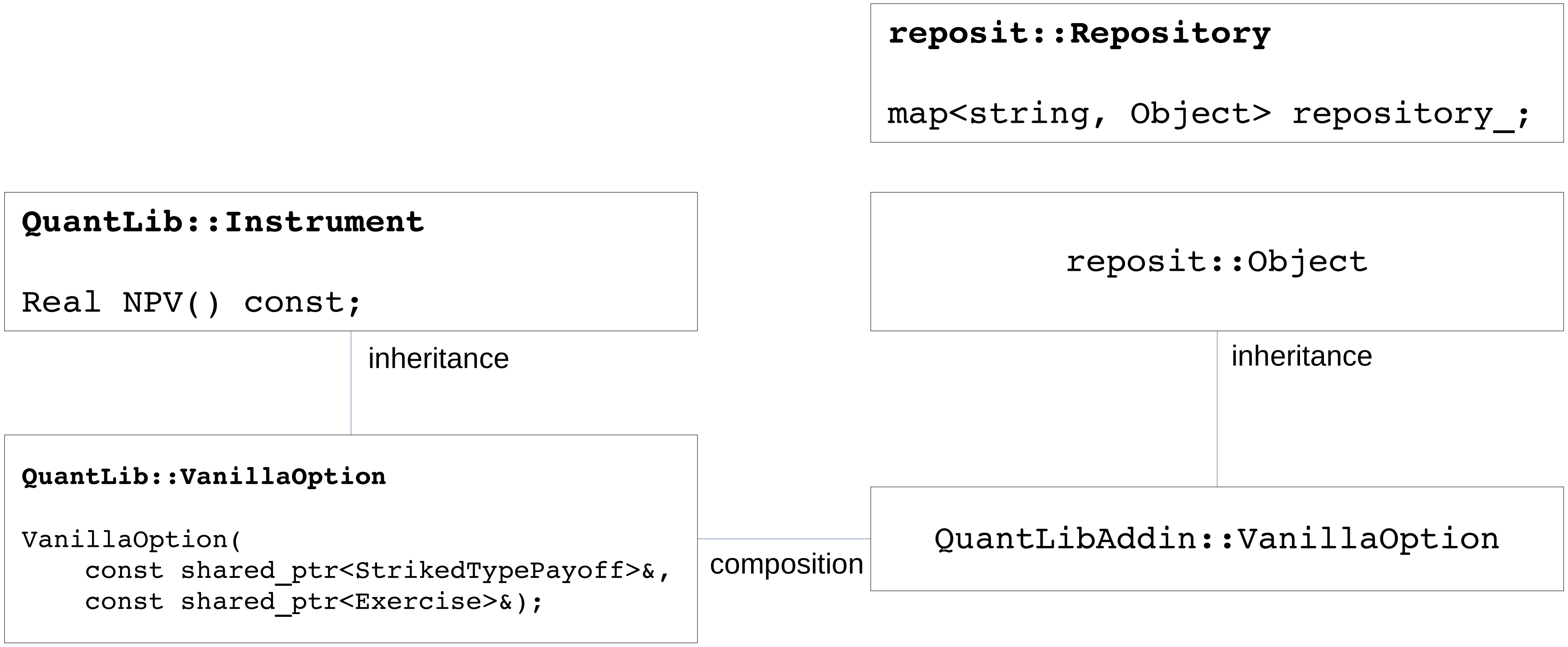
reposit Object Repository

Consider this simplified fragment of the QuantLib interface:

```
namespace QuantLib {

    //! Abstract instrument class
    /*! This class is purely abstract and defines the interface of concrete
        instruments which will be derived from this one.
    */
    class Instrument {
      public:
        //! returns the net present value of the instrument.
        Real NPV() const;
    };

    //! Vanilla option (no discrete dividends, no barriers) on a single asset
    class VanillaOption : public Instrument {
      public:
        VanillaOption(const boost::shared_ptr<StrikedTypePayoff>&,
                      const boost::shared_ptr<Exercise>&);
    };
}
```

How would you export that functionality to Excel?
How do you call a C++ constructor from Excel?
After you create the object, where do you store it?

reposit Object Repository



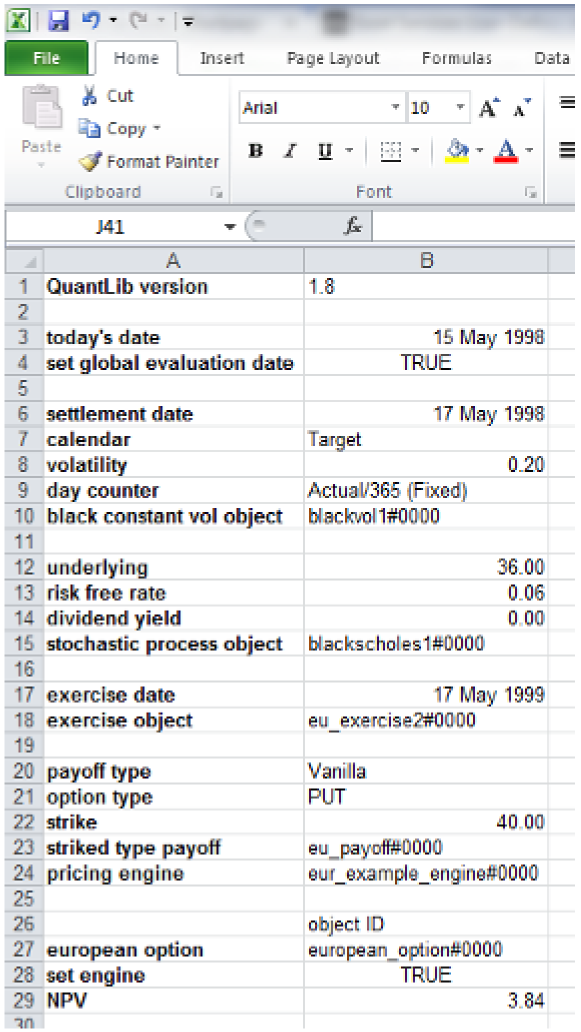
```
VanillaOption(
    const shared_ptr<StrikedTypePayoff>&,
    const shared_ptr<Exercise>&);
```

Excel worksheet functions:

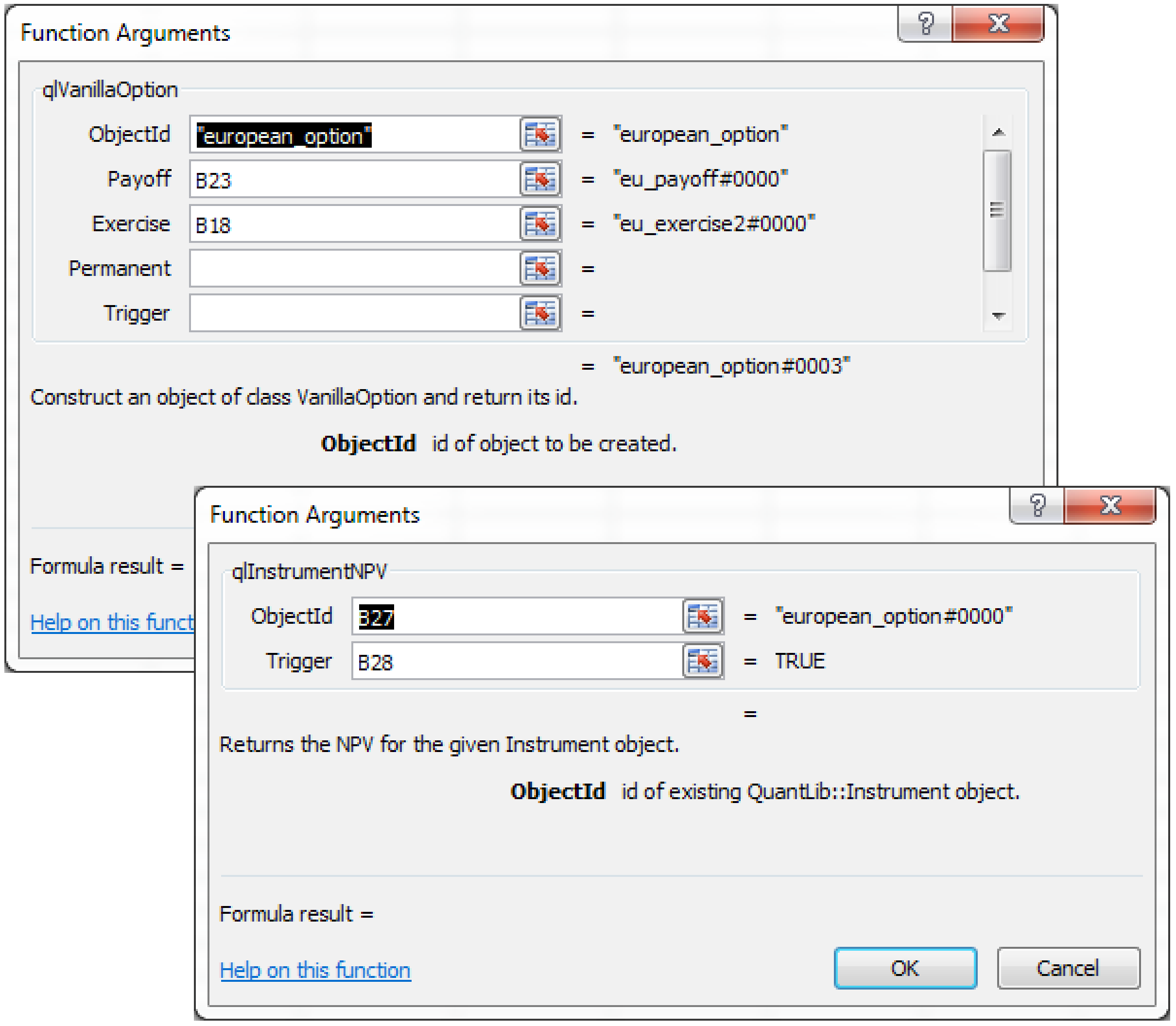
qlVanillaOption(): construct an object of type QuantLib::VanillaOption, wrap it in a QuantLibAddin::VanillaOption, and store it in the object cache.

qlInstrumentNPV(): retrieve an object from the repository, downcast it to a QuantLibAddin::Instrument, invoke member function NPV on the contained QuantLib::Instrument, and return the result to Excel.

reposit Object Repository



	A	B
1	QuantLib version	1.8
2		
3	today's date	15 May 1998
4	set global evaluation date	TRUE
5		
6	settlement date	17 May 1998
7	calendar	Target
8	volatility	0.20
9	day counter	Actual/365 (Fixed)
10	black constant vol object	blackvol1#0000
11		
12	underlying	36.00
13	risk free rate	0.06
14	dividend yield	0.00
15	stochastic process object	blackscholes1#0000
16		
17	exercise date	17 May 1999
18	exercise object	eu_exercise2#0000
19		
20	payoff type	Vanilla
21	option type	PUT
22	strike	40.00
23	striked type payoff	eu_payoff#0000
24	pricing engine	eur_example_engine#0000
25		
26		object ID
27	european option	european_option#0000
28	set engine	TRUE
29	NPV	3.84



Function Arguments

qlVanillaOption

ObjectId	"european_option"	=	"european_option"
Payoff	B23	=	"eu_payoff#0000"
Exercise	B18	=	"eu_exercise2#0000"
Permanent		=	
Trigger		=	

= "european_option#0003"

Construct an object of class VanillaOption and return its id.

ObjectId id of object to be created.

Formula result =

Help on this funct

Function Arguments

qlInstrumentNPV

ObjectId	B27	=	"european_option#0000"
Trigger	B28	=	TRUE

=

Returns the NPV for the given Instrument object.

ObjectId id of existing QuantLib::Instrument object.

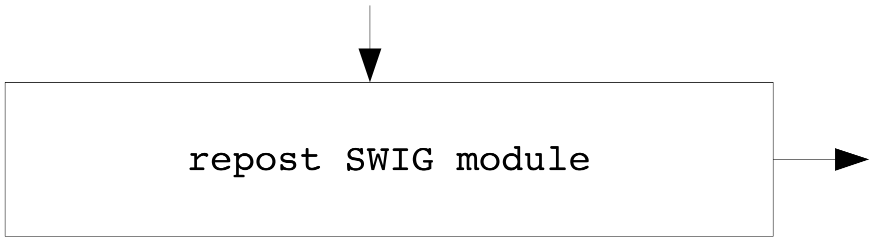
Formula result =

Help on this function

reposit SWIG Module

SWIG parses C++ header files and autogenerates source code for addins on supported target platforms.

```
namespace QuantLib {

    class Instrument {
      public:
        //! returns the net present value of the instrument.
        Real NPV() const;
    };

    class VanillaOption : public Instrument {
      public:
        VanillaOption(const boost::shared_ptr<StrikedTypePayoff>&,
                      const boost::shared_ptr<Exercise>&);
    };
}
```



```
XLL_DEC double *qlInstrumentNPV(
        char *ObjectId,
        OPER *Trigger);

XLL_DEC char *qlVanillaOption(
        char *ObjectId,
        char *Payoff,
        char *Exercise,
        OPER *Permanent,
        OPER *Trigger,
        bool *Overwrite);
```

The customized reposit SWIG module parses QuantLib header files and autogenerates QuantLibXL source code.

Supported Features

Object Repository

The repository supports a variety of features including:

- **Addins**: Bindings to Excel and C++ are provided, allowing you to implement a C++ program which replicates the behavior of your spreadsheet.
- **Inheritance**: Inheritance relationships in your C++ library are preserved in the object oriented interface that is exported to Excel.
- **Serialization**: Objects in the repository may be serialized, enabling you to save and load the state of your spreadsheet.
- **Conversions**: The library supports a wide variety of conversions between C++ and Excel data types.
- **Coercions**: You can configure coercions which allow the user to enter data of different types which are automatically converted into the target data type.
- **Enumerations**: Enumerations allow the user to enter strings which are mapped to C++ enumerated types or template classes.

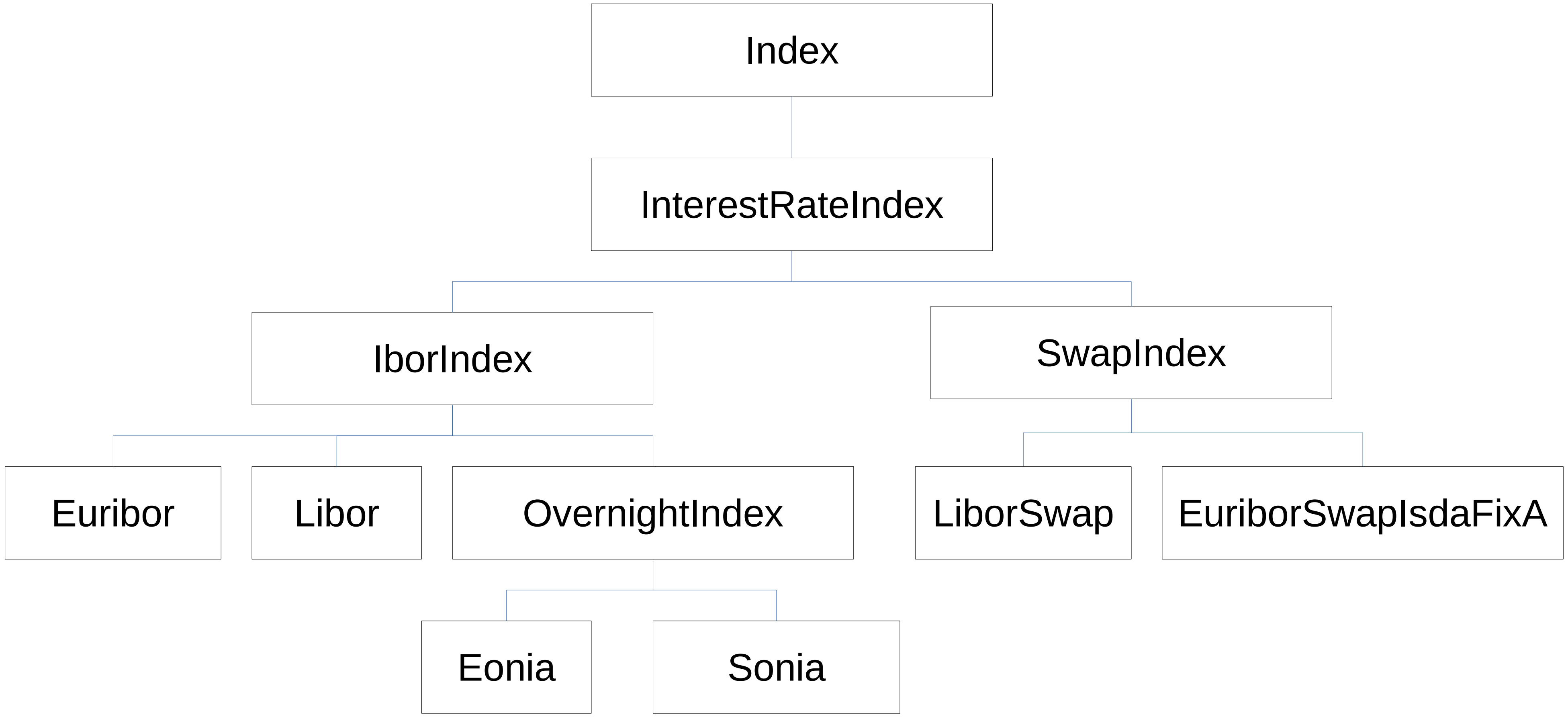
SWIG Module

Autogeneration of addin source code is provided by a customized SWIG module. The code generation utility supports the following features:

- **Functions**: Simple functions in your C++ library are easily exported to target platforms.
- **Objects**: Classes in your C++ library are exported to the C++ and Excel addins. An autogenerated interface to these classes allows end users to construct objects, store them in the repository, and query and update them.
- **Inheritance**: Object hierarchies in your C++ library are recognized by the code generation utility and preserved in the functions in the Excel/C++ addins.
- **Serialization**: All of the code required to serialize your objects is generated automatically.
- **Overrides**: A facility is provided to override the autogenerated code in case you want to customize it.

Index Class Hierarchy

A hierarchy of classes to export to Excel



In some cases we want the native QuantLib functions,
in some cases we want to override behavior in the QuantLibAddin namespace.

SWIG Interface File

Grab some functionality from the QuantLib namespace...

```
namespace QuantLib {

    class Index {
        public:
            //! Returns the fixing for the given Index object. The fixing is retrieved from the ...
            double fixing(
                const Date& fixingDate,     //!< fixing date(s).
                bool forecastTodaysFixing   //!< If set to TRUE it forces the forecasting ...
            );
            //! Returns the calendar (e.g. TARGET) for the given Index object.
            Calendar fixingCalendar();
    };

    class InterestRateIndex : public Index {
        public:
            //! Returns the fixing days (e.g. 2) for the given InterestRateIndex object.
            Natural fixingDays();
            %generate(c#, dayCounter);
            //! Returns the DayCounter (e.g. Actual/360) for the given InterestRateIndex object.
            const DayCounter& dayCounter();
            //! Returns the value date for the given fixing date for the given ...
            Date valueDate(const Date& fixingDate);
            //! Returns the tenor (i.e. length, e.g. 6M, 10Y) for the given ...
            Period tenor();
    };

}
```

SWIG Interface File

Grab some functionality from the QuantLibAddin namespace...

```
namespace QuantLibAddin {

    class Index {
        public:
            //! Adds fixings for the given Index object.
            void addFixings(
                const std::vector<QuantLib::Date>& dates,    //!< fixing dates.
                const std::vector<QuantLib::Real>& values,   //!< fixing values.
                bool forceOverwrite                          //!< Set to TRUE to force overwriting...
            );
    };

    class InterestRateIndex : public Index {};

    class IborIndex : public InterestRateIndex {};

    class SwapIndex : public InterestRateIndex {
        public:
            SwapIndex(
                const std::string& familyName,                                //!< index name.
                const QuantLib::Period& p,                                    //!< index tenor...
                QuantLib::Natural fixingDays,                                 //!< swap rate fixing...
                QuantLib::Currency& crr,                                      //!< Index Currency.
                const QuantLib::Calendar& calendar,                           //!< holiday calendar...
                const QuantLib::Period& fixedLegTenor,                        //!< tenor of the...
                QuantLib::BusinessDayConvention fixedLegBDC,                  //!< business day...
                const QuantLib::DayCounter& fixedLegDayCounter,               //!< day counter of the...
                const boost::shared_ptr<QuantLib::IborIndex>& index,          //!< swap's floating ibor..
                const QuantLib::Handle<QuantLib::YieldTermStructure>& disc   //!< discounting...
            );
    };
}
```

SWIG Interface File

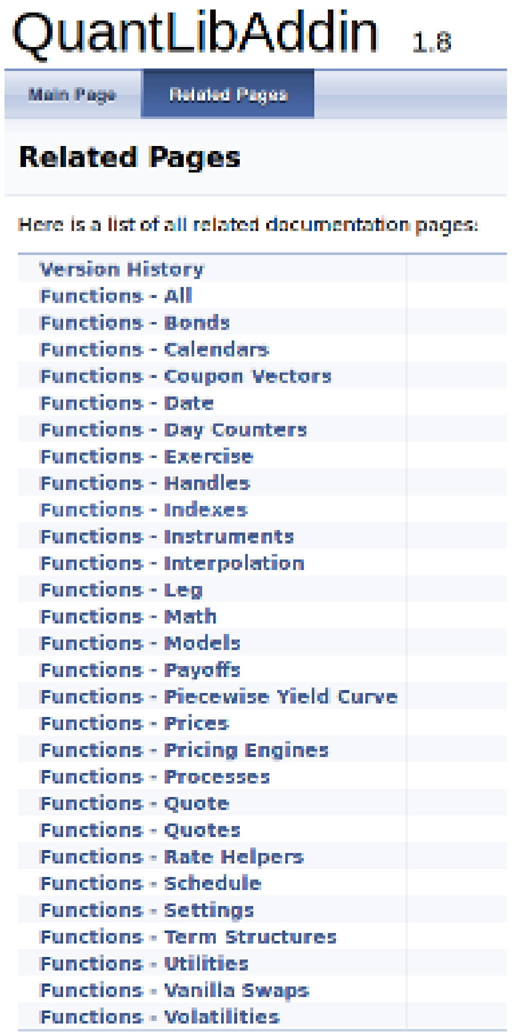
Generate the Excel addin...

```
namespace QuantLib {

    class Index {
        public:
            //! Returns the fixing for the given Index object. The fixing is retrieved from the ...
            double fixing(
                const Date& fixingDate,     //!< fixing date(s).
                bool forecastTodaysFixing   //!< If set to TRUE it forces the forecasting ...
            );
            //! Returns the calendar (e.g. TARGET) for the given Index object.
            Calendar fixingCalendar();
    };

    class InterestRateIndex : public Index {
        public:
            //! Returns the fixing days (e.g. 2) for the given InterestRateIndex object.
            Natural fixingDays();
            %generate(c#, dayCounter);
            //! Returns the DayCounter (e.g. Actual/360) for the given InterestRateIndex object.
            const DayCounter& dayCounter();
            //! Returns the value date for the given fixing date for the given ...
            Date valueDate(const Date& fixingDate);
            //! Returns the tenor (i.e. length, e.g. 6M, 10Y) for the given ...
            Period tenor();
    };

}

namespace QuantLibAddin {

    class Index {
        public:
            //! Adds fixings for the given Index object.
            void addFixings(
                const std::vector<QuantLib::Date>& dates,   //!< fixing dates.
                const std::vector<QuantLib::Real>& values,  //!< fixing values.
                bool forceOverwrite                         //!< Set to TRUE to force overwriting...
            );
    };

    class InterestRateIndex : public Index {};

    class IborIndex : public InterestRateIndex {};

    class SwapIndex : public InterestRateIndex {
        public:
            SwapIndex(
                const std::string& familyName,                              //!< index name.
                const QuantLib::Period& p,                                  //!< index tenor...
                QuantLib::Natural fixingDays,                               //!< swap rate fixing...
                QuantLib::Currency& crr,                                    //!< Index Currency.
                const QuantLib::Calendar& calendar,                         //!< holiday calendar...
                const QuantLib::Period& fixedLegTenor,                      //!< tenor of the...
                QuantLib::BusinessDayConvention fixedLegBDC,                //!< business day...
                const QuantLib::DayCounter& fixedLegDayCounter,             //!< day counter of the...
                const boost::shared_ptr<QuantLib::IborIndex>& index,        //!< swap's floating ibor..
                const QuantLib::Handle<QuantLib::YieldTermStructure>& disc  //!< discounting...
            );
    };
}
```

qlEonia()
qlEuribor()
qlEuriborSwapIsdaFixA()
qlIborIndexBusinessDayConvention()
qlIborIndexEndOfMonth()
qlIndexAddFixings()
qlIndexFixing()
qlIndexFixingCalendar()
qlInterestRateIndexDayCounter()
qlInterestRateIndexFixingDays()
qlInterestRateIndexTenor()
qlInterestRateIndexValueDate()
qlLibor()
qlLiborSwap()
qlSonia()
qlSwapIndex()

Documentation

Addin documentation is autogenerated



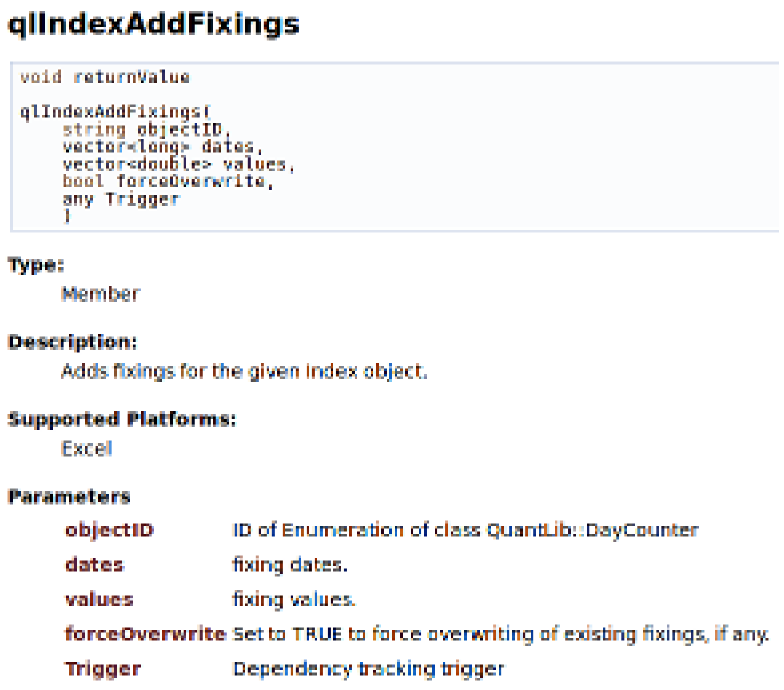
QuantLibAddin 1.8

Main Page | Related Pages

Related Pages

Here is a list of all related documentation pages:

- Version History
- Functions - All
- Functions - Bonds
- Functions - Calendars
- Functions - Coupon Vectors
- Functions - Date
- Functions - Day Counters
- Functions - Exercise
- Functions - Handles
- Functions - Indexes
- Functions - Instruments
- Functions - Interpolation
- Functions - Leg
- Functions - Math
- Functions - Models
- Functions - Payoffs
- Functions - Piecewise Yield Curve
- Functions - Prices
- Functions - Pricing Engines
- Functions - Processes
- Functions - Quote
- Functions - Quotes
- Functions - Rate Helpers
- Functions - Schedule
- Functions - Settings
- Functions - Term Structures
- Functions - Utilities
- Functions - Vanilla Swaps
- Functions - Volatilities



qlIndexAddFixings

```
void returnValue

qlIndexAddFixings(
    string objectID,
    vector<long> dates,
    vector<double> values,
    bool forceOverwrite,
    any Trigger
    )
```

Type:
Member

Description:
Adds fixings for the given index object.

Supported Platforms:
Excel

Parameters

objectID	ID of Enumeration of class QuantLib::DayCounter
dates	fixing dates.
values	fixing values.
forceOverwrite	Set to TRUE to force overwriting of existing fixings, if any.
Trigger	Dependency tracking trigger

Documentation

By default, the documentation contains no docstrings, only the names and types of functions and parameters.

```
namespace QuantLib {

    //! purely virtual base class for indexes
    /*! \warning this class performs no check that the
                 provided/requested fixings are for dates in the past,
                 i.e. for dates less than or equal to the evaluation
                 date. It is up to the client code to take care of
                 possible inconsistencies due to "seeing in the
                 future"
    */
    class Index : public Observable {
      public:
        virtual ~Index() {}
        //! Returns the name of the index.
        /*! \warning This method is used for output and comparison
                     between indexes. It is <b>not</b> meant to be
                     used for writing switch-on-type code.
        */
        virtual std::string name() const = 0;
...
```

If docstrings are present in the QuantLib header files, or the QuantLibAddin header files, or the SWIG interface files, these are included in the autogenerated documentation.

Development Status

**reposit is a work in progress
and is intended to replace an older build of QuantLibXL
in which addin source code was autogenerated by a Python script.**

Feature	**Old Build**	**New Build**
Number of Addin Functions Supported:	1,080	250
Support for Rate Curve Framework:	✓	✓
Code Autogeneration:		
Object Wrappers	X	✓
Addin Functions	✓	✓
Looping Functions	✓	✓
Enumerations	✓	X
Documentation	✓	✓
Platforms Supported:		
C++	✓	✓
Excel	✓	✓
LibreOffice Calc	✓	X
C#	X	✓

TODO List

Task	Status	Required to Supercede Old Build
remaining 750 functions	In Progress	Yes
enable support for the Excel Wizard	Pending	Yes
dynamic XLLs	Pending	Yes
add support for VC10-14	Pending	Yes
autogeneration of source code for enumerations		No
refactor conversion code and typemaps		No
re-enable the LibreOffice Spreadsheet addin		No

Table of Contents

History of Excel Versions

Version	Version	Release	Major
Number	Name	Year	Feature
2	Excel 2.0 (1987)	1987	
3	Excel 3.0 (1990)	1990	
4	Excel 4.0 (1992)	1992	C API (XLOPERs)
5	Excel 5.0 (1993)	1993	Excel Basic
7	Excel 95 (v7.0)	1995	32-bit
8	Excel 97 (v8.0)	1997	Excel VBA
9	Excel 2000 (v9.0)	2000	
10	Excel 2002 (v10.0)	2002	
11	Excel 2003 (v11.0)	2003	
12	Excel 2007 (v12.0)	2007	big grid, multithreading, ribbon
14	Excel 2010 (v14.0)	2010	64-bit
15	Excel 2013 (v15.0)	2013	
16	Excel 2016 (v16.0)	2016	

History of QuantLibXL Releases

Version	Release	Excel Feature	Calc
Number	Month & Year	Support	Addin
0.3.10	Jul 2005		Eric Ehlers
0.3.11	Oct 2005		
0.3.12	Mar 2006		
0.3.13	Aug 2006		
0.3.14	Dec 2006		
0.4.0	Feb 2007		
0.8.0	May 2007		
0.9.0	Jan 2008		
0.9.6	Sep 2008		
0.9.7	Nov 2008		
1.0.1	Oct 2010		Roland Lichters
1.1.0	May 2011		
1.2.0	Jul 2012		
1.4.0	Jun 2014	64-bit (XLL only)	
1.5.0	Apr 2015		
1.6.0	Aug 2015		Lars Callenbach
1.7.0	Dec 2015		
1.8.0	??? 2016	64-bit also for VBA	

The Future of QuantLibXL

QuantLibXL celebrates its 11th birthday this month.

Where will the project be 11 years from now?

- Will spreadsheets still exist in 2027?
- Will the QuantLibXL project last that long?
- If the QuantLibXL does last another 11 years, will it still be recognizable then, or will it have to transform into something else?

Defining Features of QuantLibXL

As we take a tour of the next generation of spreadsheets, keep in mind the chief components of QuantLibXL:

- **UDFs:** User Defined Functions. `qlSwap(), qlInstrumentNPV()`, etc.

  At present, the new generation of cloud-enabled spreadsheets do not support UDFs. This will probably have to change?

- **Repository**: A cache of objects. Currently required in order to use an object-oriented library (such as QuantLib) in a functional environment (such as Excel).

  In the new world of cloud-enabled spreadsheets, it won't make sense to cache objects in the memory of a local XLL.

- **Macros**: The Rate Curve Framework is an application layer written around QuantLibXL, in Excel VBA.

  Next generation spreadsheets offer very limited support for macros and instead provide other means for extending the UI e.g. apps and services.

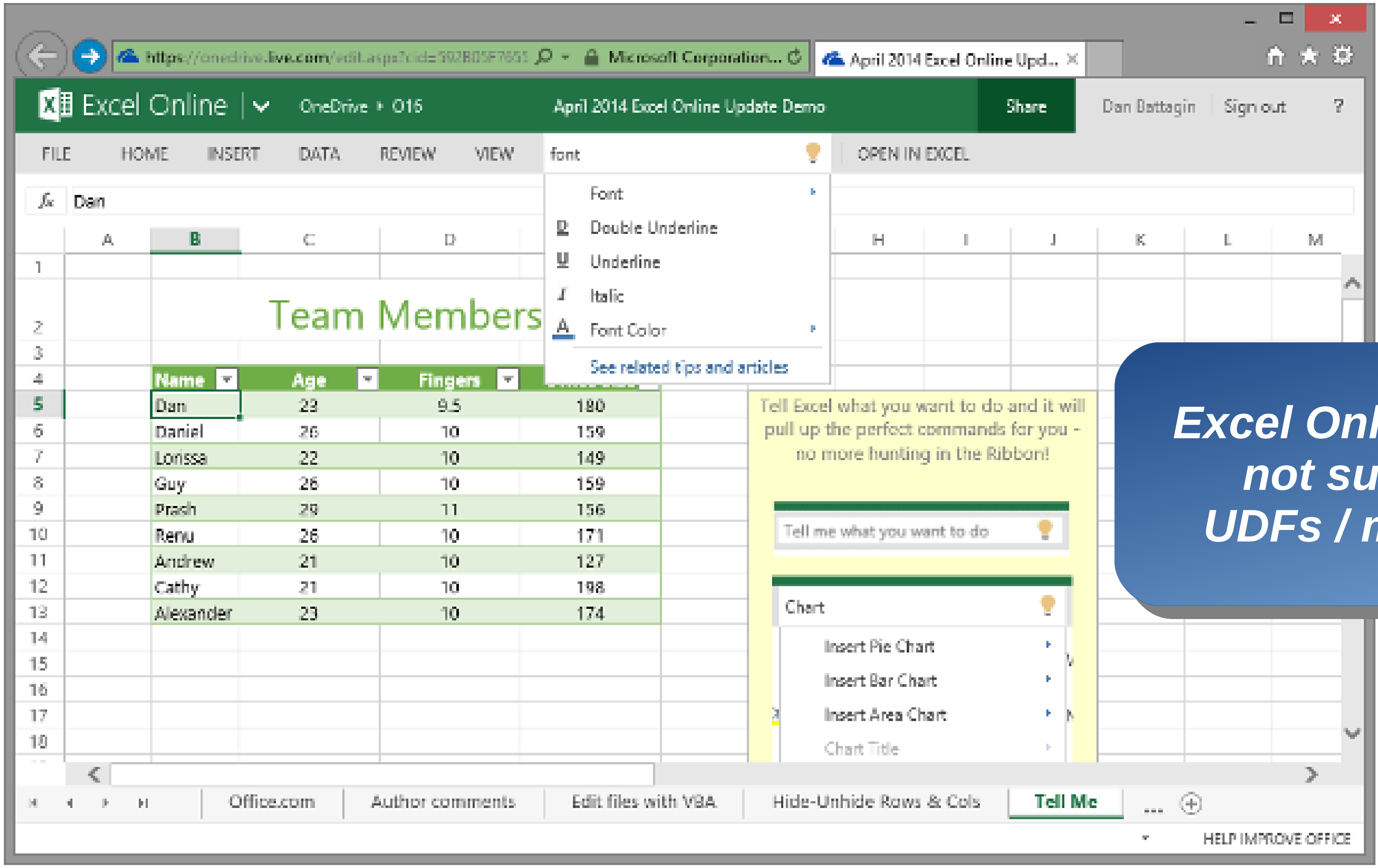
Office 365

Subscription based software-as-a-service providing rolling releases of an Office implementation that is cloud-enabled:

- OneDrive (cloud storage)
- Skype (VoIP)
- Exchange Server (calendar and mail)
- Office Online (see below)
- SharePoint (see below)

Office Online

Browser-based, lightweight implementation of Office applications.
Supports real-time co-authoring of Office documents.



Team Members

Name	Age	Fingers	
Dan	23	9.5	180
Daniel	26	10	159
Lorissa	22	10	149
Guy	26	10	159
Prash	29	11	156
Renu	26	10	171
Andrew	21	10	127
Cathy	21	10	198
Alexander	23	10	174

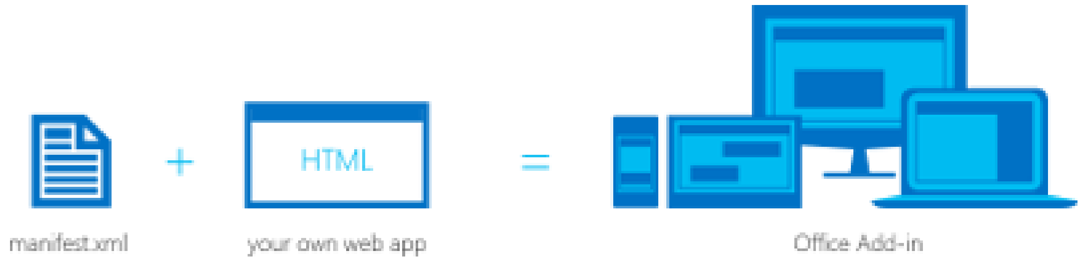
Tell Excel what you want to do and it will pull up the perfect commands for you - no more hunting in the Ribbon!

Excel Online does not support UDFs / macros.

Office Add-Ins

**Customize Office using JavaScript, HTML, and CSS -
add new functionality and content**

- Targets both desktop Excel (2013 and 2016) and Excel Online
- Extend Excel UI
- Implement interactive content via HTML and JavaScript (jQuery, Angular, etc...)
- Connect to REST endpoints and consume web services
- Invoke Excel's JavaScript API



***Office addins are hosted web services.
They do not currently support Excel UDFs.***

SharePoint

Content management platform for Office server

- Enterprise Content and Document Management
- Personal Cloud
- Intranet & Corporate Social Network
- Software Framework

SharePoint 2007 introduced Excel Services which facilitates multiuser access to spreadsheets in SharePoint Server.

SharePoint 2010 introduced the possibility to implement Excel UDFs using managed code in Excel Services.

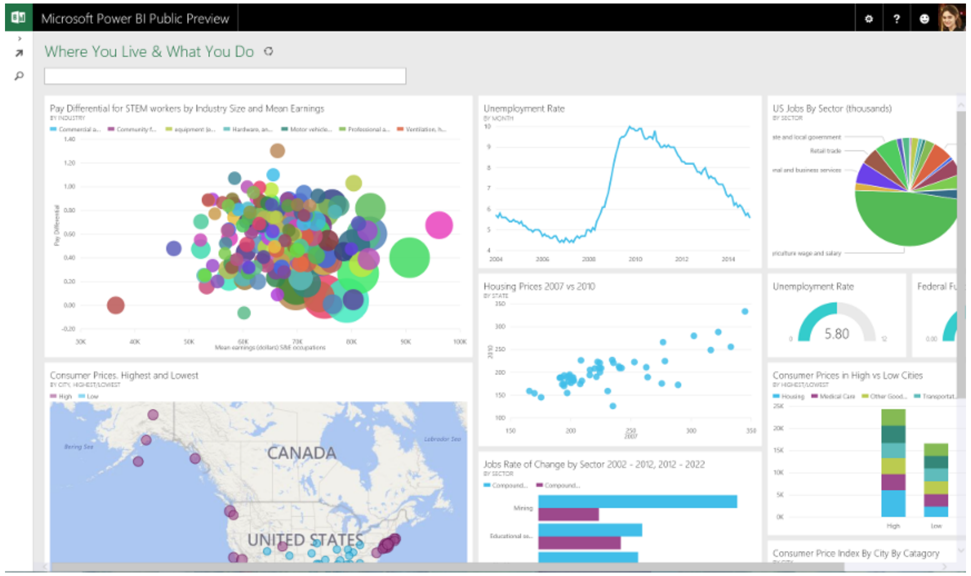
SharePoint 2013 introduced the possibility to implement Excel UDFs in JavaScript for Excel Online.

SharePoint is one of the few platforms currently supporting UDFs for cloud-enabled spreadsheets. As a proprietary enterprise solution, SharePoint is not an ideal target platform for an open source project such as QuantLibXL.

Power BI

For the sake of completeness we mention Power BI:

Microsoft proprietary business analytics service, including a series of addins for Excel capable of processing huge datasets (millions of rows)


Microsoft Power BI Public Preview
Where You Live & What You Do
Pay Differential for STEM workers by Industry Size and Mean Earnings
Unemployment Rate
US Jobs By Sector (thousands)
Housing Prices 2007 vs 2010
Unemployment Rate
5.80
Consumer Prices, Highest and Lowest
CANADA
UNITED STATES
Jobs Rate of Change by Sector 2002 - 2012, 2012 - 2022
Consumer Prices in High vs Low Cities
Consumer Price Index By City By Catagory

- **Power Query**: retrieve data from selected sources & transform it
- **Power Pivot**: merge data from multiple sources for analysis in pivot tables
- **Power View**: provides interactive visualization of PowerPivot data models
- **Power Map**: geographic representation of data

This is more about mining big data than it is about quantitative finance, and if QuantLibXL were to fit in to the above puzzle it would be at a lower level.

Google Drive

Cloud storage which supports sharing and editing of documents (Docs/Sheets/Slides).

How would you open a QuantLibXL spreadsheet in Google Drive?

Office documents can be opened on Google Drive:

- Using Office Compatibility Mode
- By converting Office documents to Google Drive format
- By installing the Google Drive plugin for Office

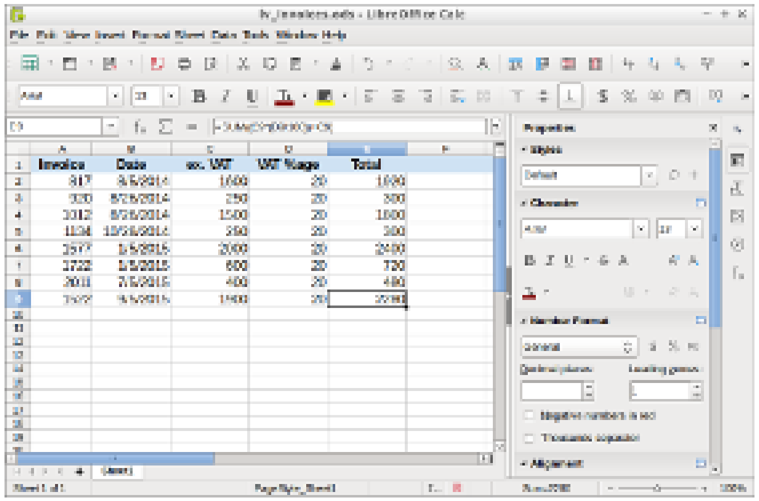
Supports Google Apps Script but not Excel macros/addins.

It would take a lot of development effort to get a QuantLibXL spreadsheet working on Google Drive.

Libre Office Online (LOOL)

A web server implementing an HTML5 UI which replicates the behavior of LibreOffice

- Promises complete fidelity between LibreOffice desktop and LibreOffice Online
- All Writer, Calc, and Impress supported file-types supported
- Open source project which can be hosted on private hardware



Invoice	Date	ex. VAT	VAT %age	Total
817	8/5/2014	1600	20	1920
920	8/26/2014	250	20	300
1012	8/28/2014	1500	20	1800
1134	10/28/2014	250	20	300
1577	1/5/2015	2000	20	2400
1722	1/5/2015	600	20	720
2011	7/5/2015	400	20	480
[illegible]	[illegible]	1900	20	[illegible]

Offers another possibility for implementing the QuantLibAddin interface on the cloud.

Conclusions

- As of today, the desktop version of Excel (2016) continues to support the features required by QuantLibXL (UDFs, cache, macros).
- QuantLibXL will continue to target desktop Excel for as long as that platform remains viable.
- Cloud-enabled spreadsheets do not yet support a general mechanism for implementing UDFs.
- If one day Excel Online supports UDFs, QuantLibXL could be implemented as a web service targeting Excel Online and allowing existing QuantLibXL workbooks to migrate from desktop Excel to Excel Online.
- We would need to find a new solution for object cache and macros.