Docklight Application Note: DLL Automation API

1 Overview

The **Docklight DLL Automation API** is a simple way to add Docklight-style communications (serial COM port, TCP or UDP, Send and Receive Sequences) to your **Microsoft Excel** workbook, **Windows Script** or Windows **C#, C++, C** or **Java** software application.

Supported platforms are Windows 10, Windows 8, Windows 8 x64, Windows 7, Windows 7 x64, Windows Vista, Windows Vista x64 and Windows XP.

1.1 How it works

The Docklight Automation API allows access to the full command set described in the **Docklight Scripting Manual – The DL Object** [3] (www.docklight.de/manual/dlobject.htm). You can load existing Docklight projects files using **OpenProject**, load predefined Docklight Options with **LoadProgramOptions**, and use

the same communications processing and **Send/Receive Sequence** functionality that is available in the **Docklight Scripting** desktop application.

1.2 Additional DLL Automation Methods

Besides the standard **DL Object** methods and properties, the following DLL-specific commands are available:

Function	Description
result = dlObj.**GetCommWindowData ([***representation***])** Example: hexDisplayStr = dlObj.GetCommWindowData("H")	Returns the accumulated contents of the communication windows buffer. *representation* - the window buffer and display format to use: "A" = ASCII (default), "H" = HEX, "D" = Decimal or "B" = Binary. For DLL clients there is no visible "Communication Window". Instead **GetCommWindowData** allows access to a plain text buffer with the same output. The display buffer for *representation* is cleared after each call to **GetCommWindowData**. NOTE: Only a *representation* enabled in **Docklight Options – Communication Window Modes** (http://docklight.de/manual/options.htm) can be used. By default, this is ASCII, HEX and Decimal. If required, load different options using **LoadProgramOptions.** NOTE: The maximum size of the Communication Window buffer is 128000 characters. If more communication data is accumulating without you collecting it using **GetCommWindowData**, the oldest data gets deleted.
result **=** dlObj.**GetReceiveComments()**	*result* contains all Receive Sequence Comments in chronological order, separated by a line break, since the last call of **GetReceiveComments**. Use **GetReceiveComments** and Receive Sequence comment macros to implement a parser for all incoming Receive Sequence data, as an alternative to the **OnReceive()** event procedures [4] available in Docklight Scripting. NOTE: A maximum of 10000 Receive Sequence events are stored and returned by **GetReceiveComments**, which should be sufficient for all practical applications.
result **=** dlObj.**DllSetLicense(** *licenseKey***,** *registeredUser***)** Example:	Deactivates the evaluation messages using your individual license key and the registered name, as listed in your original delivery e-mail and delivery note. *result* is True, if *licenseKey* was accepted, False if the license key is invalid, does not belong to this DLL Automation

success = dlObj.DllSetLicense("01 04 1234567", "My Company")	package, or the *registeredUser* does not match the license key.
dlObj.**PumpCommData()**	Processes and formats the accumulated communication data and creates Receive Sequence events. Only required when using the **DocklightFunctions.dll** C/C++ API directly. (See section 1.4 DLL API Packages and section 3.2 C++ Win32 Console App Example.) NOTE: The **Pause(…)** method will achieve the same task, but also adds a specified delay for program execution. Calling **PumpCommData** is equivalent to calling **Pause(0)**.
result **=** dlObj. **GetDocklightErrorNo()**	Returns the error number of the last DL object call, or 0 (zero), if the last call was successful. NOTE: For error messages generated by Docklight, and not general VB Script errors, a fixed error number of vbObjectError + 1100 is used. See the Microsoft documentation on vbObjectError.
result **=** dlObj. **GetDocklightErrorDescription()**	Returns the corresponding error description text.

1.3 Limitations and Differences

Compared to a Docklight Script (**.pts** file) executed by the Docklight Scripting desktop application, the Docklight Automation API is different in the following ways:

- You cannot use the Docklight Scripting **OnSend** / **OnReceive** event procedures [4], since they are part of the Docklight Scripting script engine. Use the **GetReceiveComments** and/or **WaitForSequence** methods instead and roll your own multithreading in your program code, if required.
- **DL.Quit** does not exist.
- Communication Driver **Mode: External / High Priority** [5] is not available.
- The DL Object method **GetEnvironmentVariable** was renamed to **GetEnvironment** for the Docklight Automation API, to avoid confusion with the Windows API function of the same name. (Future versions of Docklight Scripting will also use the GetEnvironment name, while still supporting the old name for compatibility.)
- When using the **DocklightFunctions.dll** API (see section 3.1), you need to take care of Windows messaging and Docklight communications processing by either calling **DL.PumpCommData()** or **DL.Pause()** in regular short-term intervals.
- Even though the API code allows creating and using **more than one DL object** within one host application, it is currently **not supported**, i.e. the behavior is undefined.

1.4 DLL API Packages

- **DocklightAutomation.dll** – Docklight Automation Object Library (**OLE** / **ActiveX** / **COM** Library)

  This is the easiest way to use Docklight from within Microsoft **Word** or Microsoft **Excel** macros written in **VBA** (Visual Basic for Applications), for creating **VBScript** code to run on your computer, or for any programming framework that allows accessing COM objects, including **C#/.NET** and **Java** (using com4j [8], JACOB [9] or any other third-party COM bridge).

  NOTE: COM in the above section refers to the Microsoft Component Object Model [7], not a Windows serial COM port.

- **DocklightFunctions.dll** – Docklight Function Library

  A classic C/C++-style DLL accompanied by a header file (DocklightFunctions.h) and linker LIB file (DocklightFunctions.lib). This is the way to go for **C** or **C++** programs, including **Qt/C++** applications, **Labview**, or any other software that allows importing 32 bit DLLs with standard C function calls and data types.

1.5 Evaluation Mode, Licensing and Redistribution

The Docklight Automation DLLs can be freely distributed and used for evaluation purposes. In evaluation / unregistered mode, the full functionality is available, but additional evaluation messages will appear in both TX and RX data directions:

```
Docklight DLL evaluation, for purchase please visit www.docklight.de
```

After purchasing a Docklight Automation DLL license key, please add the command

```
DllSetLicense "<your license key>", "<your registered user name>"
```

in your program code. See the VBScript example in section 2.2.

Redistribution of the Docklight DLL license key along with your own works is possible, provided that the conditions of the Docklight DLL License [6] (www.docklight.de/pdf/fuh_dll_license_int_web.txt) are met.

2 DocklightAutomation.dll

2.1 How to Use

- Download the Docklight Automation DLL package from the following link:

  www.docklight.de/download/Docklight_Dll.zip [1]

- Copy the **DocklightAutomation.dll** and **DckFctLb.dll** files to a location on your computer that will not change.

- Register **DocklightAutomation.dll**, e.g. run a **cmd.exe** command window as **Administrator**, go to the location of the DocklightAutomation.dll and type:

  ```
  regsvr32 DocklightAutomation.dll
  ```

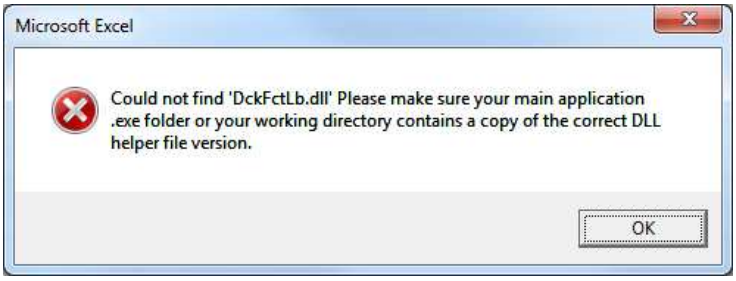
  (You can use the file **DocklightAutomation_register_RunAsAdministrator.bat** from the DLL package for this. Right-click and choose **Run as administrator**.)

You can unregister DocklightAutomation.dll using

```
regsvr32 –u DocklightAutomation.dll
```

- Include a copy of **DckFctLb.dll** in your application .exe folder, or in your current working folder, e.g. the folder of your Excel workbook.
- Create a **DocklightAutomation.DL** COM object in your application.
- The new object offers the methods and properties from the **Docklight Scripting DL Object** [3] (www.docklight.de/manual/dlobject.htm), plus the DLL-specific additions described in section 1.2.

NOTE: For the **DckFctLb.dll** helper DLL, Windows will not check the **DocklightAutomation.dll** folder, but instead the location of your client executable application (e.g. Excel.exe) and your current application folder (working folder).



To fix this, copy **DckFctLb.dll** to your application working folder, e.g. the location of your Excel workbook.

It is mandatory to check the **GetDocklightErrorNo()** value directly after creating a **DL** object. Any value different from 0 would indicate a DLL initialization error. A detailed error message can be obtained using **GetDocklightErrorDescription()**, as shown in the later examples.

2.2 VBScript Example

```
' Requirements:
'    regsvr32 DocklightAutomation.dll
' How to call from command line:
'    cscript DocklightAutomation_Sample_VBScript.vbs
' or on Windows x64 operating systems:
'    %windir%\SysWOW64\cscript.exe DocklightAutomation_Sample_VBScript.vbs
'    (explicit call to the 32 bit version of cscript required)

Set dlObj = CreateObject("DocklightAutomation.DL")

' ====================================================
' Docklight DLL is Copyright Flachmann und Heggelbacher GbR, www.docklight.de.
' Redistribution Runtime License. Development or modification of applications
' is only allowed for the registered user below.
dlObj.DllSetLicense "<YOUR LICENSE KEY>", "<YOUR REGISTERED USER NAME>"
' ====================================================

dlObj.OpenProject "PingPong_UDP_ReceiveComments.ptp"
' How to check for errors (i.e. "could not open project file")
If dlObj.GetDocklightErrorNo() <> 0 Then
    WScript.StdOut.WriteLine(dlObj.GetDocklightErrorDescription())
    WScript.Stop
```

```
End If
' Generate small amount of communication data using UDP loopback
dlObj.StartCommunication
dlObj.SendSequence "Pong"
' do this for a little while
For i = 1 to 1000
    dlObj.PumpCommData()
    ' and read the accumulated Receive Sequence Comments
    myReceiveComments = dlObj.GetReceiveComments()
    ' Every Receive Sequence event is a single comment line
    outputLines = Split(myReceiveComments, vbCrLf)
    For Each line In outputLines
         If Len(line) > 0 Then
              WScript.StdOut.WriteLine(line)
         End If
    Next
Next
```

2.3 Excel Macro (VBA) Example

Using the **DocklightAutomation.DL** COM object in Microsoft Excel is very similar to VBScript. Make sure the correct file type is used: **Excel Macro-Enabled Workbook (.xslm)**.

An example file, **DocklightAutomation_Sample_Excel.xlsm**, is included in the Docklight DLL package. To view the VBA code, go to the **View** toolbar, click on the **Macros** element on the right side and select **View Macros**. The code used in the included example looks like this:

```
' DocklightAutomation_Sample_Excel.xlsm
' Author: Oliver Heggelbacher
' Applies To: Docklight Automation Object Library / DocklightAutomation.dll
'
' Demonstrates how to use an existing Docklight project
' for performing a serial communication task, and directly
' evaluate and process the results in Excel.
'
' It uses a simplistic UDP loopback and PingPong example, so no
' additional communication equipment is required.

Sub DocklightTest()
    ' This makes sure that dlObj.OpenProject will find the .ptp file,
    ' and the DocklightAutomation.DL object will find the DckFctLb.dll helper file.
    ChDir ThisWorkbook.Path
    ChDrive ThisWorkbook.Path

    ' Start with clean display
    Rows("9:999").Select
    Selection.ClearContents
    currentRow = 9

    ' Requires that DocklightAutomation.dll is registered ("regsvr32
DocklightAutomation.dll")
    ' and a copy of the DckFctLb.dll is in the same folder as the Excel workbook
    Set dlObj = CreateObject("DocklightAutomation.DL")
    ' Checking for errors here is mandatory
```

```
    If dlObj.GetDocklightErrorNo() <> 0 Then
        MsgBox (dlObj.GetDocklightErrorDescription()), vbCritical + vbOKOnly
        End
    End If

    ' make sure default Docklight Options are used: date/time format, representations
used, a.s.o...
    dlObj.LoadProgramOptions "DocklightStandardOptions.xml"
    If dlObj.GetDocklightErrorNo() <> 0 Then
        MsgBox (dlObj.GetDocklightErrorDescription())
    End If

    ' Create some sample communication output data using a local UDP loopback
    dlObj.OpenProject "PingPong_UDP_Loopback.ptp"
    If dlObj.GetDocklightErrorNo() <> 0 Then
        MsgBox (dlObj.GetDocklightErrorDescription())
    End If

    dlObj.StartCommunication
    dlObj.SendSequence "Pong"
    dlObj.Pause 20

    ' now batch-analyze the output
    myCommWindowOutput = dlObj.GetCommWindowData("A")
    splitToLines = Split(myCommWindowOutput, vbCrLf)
    For Each nextLine In splitToLines
        isTX = InStr(nextLine, "[TX]")
        isRX = InStr(nextLine, "[RX]")
        If isTX Or isRX Then
            ' column A is the time stamp, not including the milliseconds
            Range("A" + CStr(currentRow)).Select
            dirStartPos = InStr(nextLine, "[")
            ActiveCell.FormulaR1C1 = Left(nextLine, dirStartPos - 6)
            ' milliseconds
            Range("B" + CStr(currentRow)).Select
            ActiveCell.FormulaR1C1 = Mid(nextLine, dirStartPos - 4, 3)
            ' Tthe direction column: TX or RX
            Range("C" + CStr(currentRow)).Select
            dirEndPos = InStr(nextLine, "]")
            ActiveCell.FormulaR1C1 = Mid(nextLine, dirStartPos + 1, dirEndPos -
dirStartPos - 1)
            ' rest of the line is the actual data
            Range("D" + CStr(currentRow)).Select
            ActiveCell.FormulaR1C1 = Mid(nextLine, dirEndPos + 3)
            ' let's make this colorful
            ' (the color codes come from what the macro recorder indicates for blue
and red Excel standard colors)
            ActiveCell.Font.Color = IIf(isTX, -4165632, -16777024)
            ' next Excel table row
            currentRow = currentRow + 1
        End If
    Next
End Sub
```

2.4 C# Example

- Create a **C# Console Application** project in Visual Studio.
- Click on **Add Reference**, select the **COM** tab and select **Docklight Automation Object Library**.
- Enter the following code into the **Main()** function body:

```
DocklightAutomation.DL dlObj = new DocklightAutomation.DL();
dlObj.OpenProject("C:/Docklight_dll/PingPong_UDP_Loopback.ptp"); // path to
project
dlObj.StartCommunication();
dlObj.SendSequence("Pong");
dlObj.Pause(20);
string str = dlObj.GetCommWindowData("A");
Console.WriteLine(str);
Console.ReadLine();
```

- Copy the file **DckFctLb.dll** in your binary output or working folder, e.g. **\bin\Debug** or **\bin\Release**.
- Build and run your project.

2.5 Java Example

Accessing the DocklightAutomation COM objects requires a third-party COM bridge component, like com4j [8] or JACOB [9].

The following example and the source code provided along with Docklight DLL package uses **com4j** [8] http://com4j.kohsuke.org/tutorial.html) and **Eclipse for Java Developers** [14].

- Make sure your Java project folder contains the following files, as provided in the Docklight DLL **DocklightAutomationJavaExample** folder:
  `com4j.jar`
  `com4j-x86.dll`
  `com4j-amd64.dll`
  `com4j_LICENSE.txt`
  `DckFctLb.dll`
  `\DocklightAutomation\_DL.java`
  `\DocklightAutomation\ClassFactory.java`
- Open your Java project in Eclipse, and add the **com4j.jar** Library:
  Go to the Project **Properties -> Java Build Path -> Libraries** and select **Add JARs…**

TIP: In the Eclipse **Package Explorer**, use **Refresh / F5** to update the package view.

- Create a **Java** class that uses the Docklight functions. For example:

```
import DocklightAutomation._DL;
import java.util.Date;

public class DocklightTest {
    public static void main(String[] args) {
        _DL dlObj = DocklightAutomation.ClassFactory.createDL();
```

```
        dlObj.openProject("../PingPong_UDP_Loopback.ptp");
        // How to check for errors (i.e. "could not open project file")
        if (dlObj.getDocklightErrorNo() != 0) {
            System.out.println(dlObj.getDocklightErrorDescription());
            return;
        }

        dlObj.startCommunication();
        //! Again, checking errors
        if (dlObj.getDocklightErrorNo() != 0) {
            System.out.println(dlObj.getDocklightErrorDescription());
            return;
        }

        // You need to provide all default arguments myself in Java:
        // "A" for ASCII representation
        // (See the _DL.java file for all DL methods and their default arguments).
        dlObj.sendSequence("Pong", "", "A");

        // Accumulate a small amount of data and print
        dlObj.pause(20);
        String output_str = dlObj.getCommWindowData("A");
        System.out.println(output_str);

        // How to use dlObj.getDocklightTimeStamp
        // (again, we need to provide the default arguments)
        Date zeroTimestamp = new Date(0);
        String timeStamp = dlObj.getDocklightTimeStamp(zeroTimestamp, (short)-1,
false);
        System.out.println("Test ended at: " + timeStamp);
    }
}
```

NOTE: Java does not support optional arguments with default values. You need to specify all arguments. Check the **_DL.java** file for the default values of arguments you do not need.

NOTE: **_DL.java** and **ClassFactory.java** are the output files created by com4j [8]. To recreate these files, call com4j's **tlbimp.jar** tool from a command line prompt, e.g:

`java -jar tlbimp.jar -o output -p DocklightAutomation "..path...\DocklightAutomation.dll"`

(You will receive a warning that the OLE Date type wrapper **VT_DATE** cannot be printed. This is ok and by design.)

3 DocklightFunctions.dll

3.1 How to Use

- Download the Docklight Automation DLL package from the following link:

  www.docklight.de/download/Docklight_Dll.zip [1]

- Copy the files **DocklightFunctions.dll**, **DocklightAutomation.dll** and **DckFctLb.dll** to your application .exe folder (or to the target output folder of your C++ compiler).
- Copy the files **DocklightFunctions.h**, **DocklightFunctions.lib** and **DocklightFunctions.manifest** to a location where your compiler and linker will find them.
- Embed **DocklightFunctions.dll.manifest** into your **.exe** output file as an additional manifest. Alternatively, you can add the following **<file …>** dependency information to your existing manifest file:

```
<assembly …>
<assemblyIdentity name="MyOwnApp"…
     <file name="DocklightAutomation.dll">
          <typelib tlbid="{9E6C3C61-698E-4D20-8FC5-B7D1EF9524BC}" version="1.0"
flags="" helpdir="" />
          <comClass clsid="{545EFCF1-16D9-4999-BA53-3E53753C1F1C}"
tlbid="{545EFCF1-16D9-4999-BA53-3E53753C1F1C}" threadingModel="Apartment"
progid="DocklightAutomation.DL" description="" />
     </file>
```

- Build your application.
- To deploy your application, copy the Docklight Automation DLL files plus your own application files to a different into a folder. For example:
  `DocklightFunctions.dll`
  `DocklightAutomation.dll`
  `DckFctLb.dll`
  `MyOwnApp.exe`
  (`MyOwnApp.exe.manifest` – if you are using an external manifest file)

NOTE: This approach of building and deploying an application is what Microsoft calls “Local Deployment” [10]. **DocklightFunctions.dll** and **DckFctLb.dll** were statically linked and have no additional dependencies. The **DocklightFunctions.dll.manifest** contains the “Registration-free COM” [11] information required for **DocklightAutomation.dll**.

NOTE: As an alternative to the manifest files and “Registration-free COM”, you could use an installer that registers **DocklightAutomation.dll** on the target computer using `regsvr32 DocklightAutomation.dll`, as described in section 2.1. **DocklightFunctions.dll** itself does not require registration.

3.2 C++ Win32 Console App Example

Below is the C++ source code for a Win32 Console Application in Visual Studio 2008.

Everything inside the _**tmain** block is standard C++ and should compile on any other C++ development toolkit, including more recent version of Visual Studio [12] or Eclipse for C/C++ Developers [13].

Binary files and the full source code are provided in the Docklight Automation DLL package [1].

```
//! DocklightFunctionsCppExample.cpp - ANSI C++ example for accessing
DocklightFunctions.dll

#include "stdafx.h"
#include <iostream>
#include <assert.h>
```

```
using namespace std;

#include "DocklightFunctions.h"

int _tmain(int argc, _TCHAR* argv[])
{
    char strBuf[1024];
    //! separate buffer for the Communication Window contents.
    //! can be 128000 characters max. (extra '\0' zero termination character!)
    char commWinBuf[128001];

    DlGetCommWindowData(commWinBuf, sizeof(commWinBuf), "z");

    cout << "DocklightFunctions.dll / DocklightAutomation.dll Test " << endl;
    bool success = (DlInitialize() == 0);


    DlGetCommWindowData(commWinBuf, sizeof(commWinBuf), "z");

    if (!success) {
         //! display additional error information
         DlGetDocklightErrorDescription(strBuf, sizeof(strBuf));
         cout << "DLInitialize() Error: " << strBuf << endl;
    }
    if (success) {
         //! The ".\\" is just a reminder that if you use a path with "\",
         //! make sure you use \\ as a separator
         HRESULT retValue = DlOpenProject(".\\PingPong_UDP_Loopback.ptp");
         //! This should be always zero, unless the Docklight library was not
initialized properly at all.
         assert(retValue == 0);
         //! Display an appropriate "could not open" message, in case the project file
was not found
         if (DlGetDocklightErrorNo() != 0) {
              DlGetDocklightErrorDescription(strBuf, sizeof(strBuf));
              cout << "Error: " << strBuf << endl;
              success = false;
         }
    }
    if (success) {
         //! Generate small amount of communication data using UDP loopback
         DlStartCommunication();
         DlSendSequence("Pong");
         for (int i = 0; i < 20; i++) {
              //! need to call this in regular intervals.
              DlPumpCommData();
              //! (Use a separate worker thread for all Dl.. calls if you need true
              //! asynchronous operation)

              //! Update communication window display
              if (DlGetCommWindowData(commWinBuf, sizeof(commWinBuf), "A") > 0) {
                   cout << commWinBuf;
              }

              //! Extra display of the Receive Sequences accumulated since last time
```

```
            if (DlGetReceiveComments(commWinBuf, sizeof(commWinBuf)) > 0) {
                cout << endl << "Collected Receive Comments ===" << endl;
                cout << commWinBuf;
                cout <<          "==============================" << endl;
            }
        }
        DlStopCommunication();
    }
    DlTerminate();

    cout << endl << "Example ended. Press [ENTER] to quit." << endl;
    cin.get();
    return (success ? 0 : 1);
}
```

4 Copyright

Contact

E-Mail Support: docklight@fuh-edv.de
Flachmann und Heggelbacher
Waldkirchbogen 27
D-82061 Neuried (Munich)
Germany
http://www.fuh-edv.de

4.1 com4j Copyright Notice

The Java examples use Kohsuke Kawaguchi's com4j component. The following Copyright Notice applies:

```
OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS
BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,
OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER
IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF
THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
THE POSSIBILITY OF SUCH DAMAGE.
```

5 References

[1] Docklight DLL Automation API Download
http://docklight.de/download/Docklight_Dll.zip

[2] Docklight Scripting User Manual
http://docklight.de/pdf/docklight_scripting_manual.pdf

[3] Docklight Scripting User Manual – The DL Object
http://docklight.de/manual/dlobject.htm

[4] Docklight Scripting User Manual – OnSend / OnReceive Event Procedures
http://docklight.de/manual/dl_onsend()anddl_onreceive().htm

[5] Docklight Scripting User Manual – Communication Driver Mode: External / High Priority
http://docklight.de/manual/dialog_expert_options.htm

[6] Docklight Automation DLL License (Template)
http://docklight.de/pdf/fuh_dll_license_int_web.txt

[7] Microsoft® COM (Component Object Model)
http://www.microsoft.com/com

[8] com4j JAVA COM bridge introduction
http://com4j.kohsuke.org/tutorial.html

[9] JACOB JAVA COM Bridge on sourceforge
http://sourceforge.net/projects/jacob-project/

[10] Visual Studio: Deployment in Visual C++
http://msdn.microsoft.com/en-us/library/vstudio/dd293574

[11] Registration-Free Activation of COM Components: A Walkthrough
http://msdn.microsoft.com/en-us/library/ms973913.aspx

[12] Walkthrough: Creating a Standard C++ Program (C++)
http://msdn.microsoft.com/en-us/library/vstudio/ms235629.aspx

[13] Eclipse for C/C++ Developers
http://www.eclipse.org/callisto/c-dev.php

[14] Eclipse for Java Developers
http://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/junosr2

[15] Qt Creator – Cross platform IDE for Qt with C++, QML and ECMAscript support
http://qt-project.org/wiki/Category:Tools::QtCreator