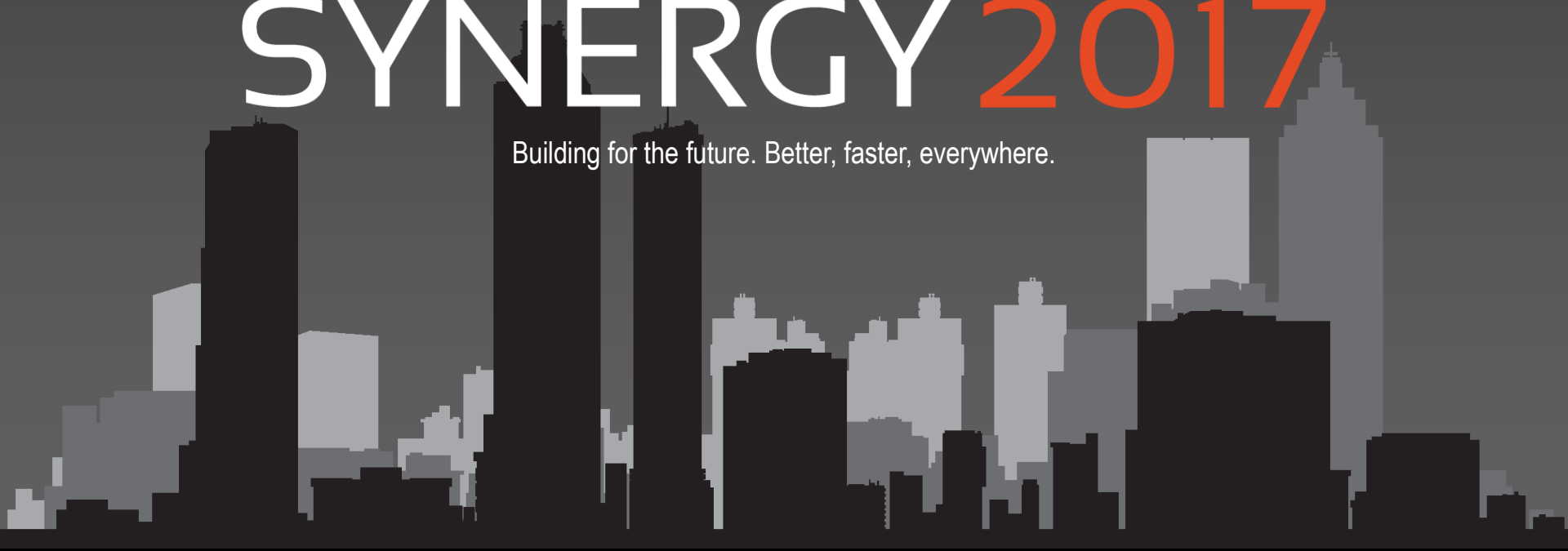
6 – 7 . APRIL ATLANTA, GA, USA

SYNERGY 2017

Building for the future. Better, faster, everywhere.



Microsoft Office Integration

Extend your Application Capabilities

Let’s say you need to…

- Create an Excel document with a picture
- Create a Word template with bookmarks
- Get access to your Outlook e-mail folders

What do you need?

- COM classes to access (one of) the products in the Office suite
- Read (understand) the Microsoft documentation

You certainly need to…

- Get over the idea that COM is a ~~misery~~ mystery
- Understand how Office products are structured
- Think like a VBA developer
  - It'll be ok; it's really not as difficult as it sounds
- Determine which version of Office you will use

Understanding the Office Structure

- Document Object Model (DOM)
- Starts from an Application object
- Contains single value and collection objects
  - E.g.: A document is a collection of paragraphs, is a collection of sentences, is a collection of words, is a collection of characters

A Quick Note about Performance

- Office products are out-of-process COM servers (self running applications that we talk to)
  - Makes accessing slower so you may want to avoid "computationally expensive" operations
    - Querying the number of words in a document feels like it takes forever…

Selecting the Office Version

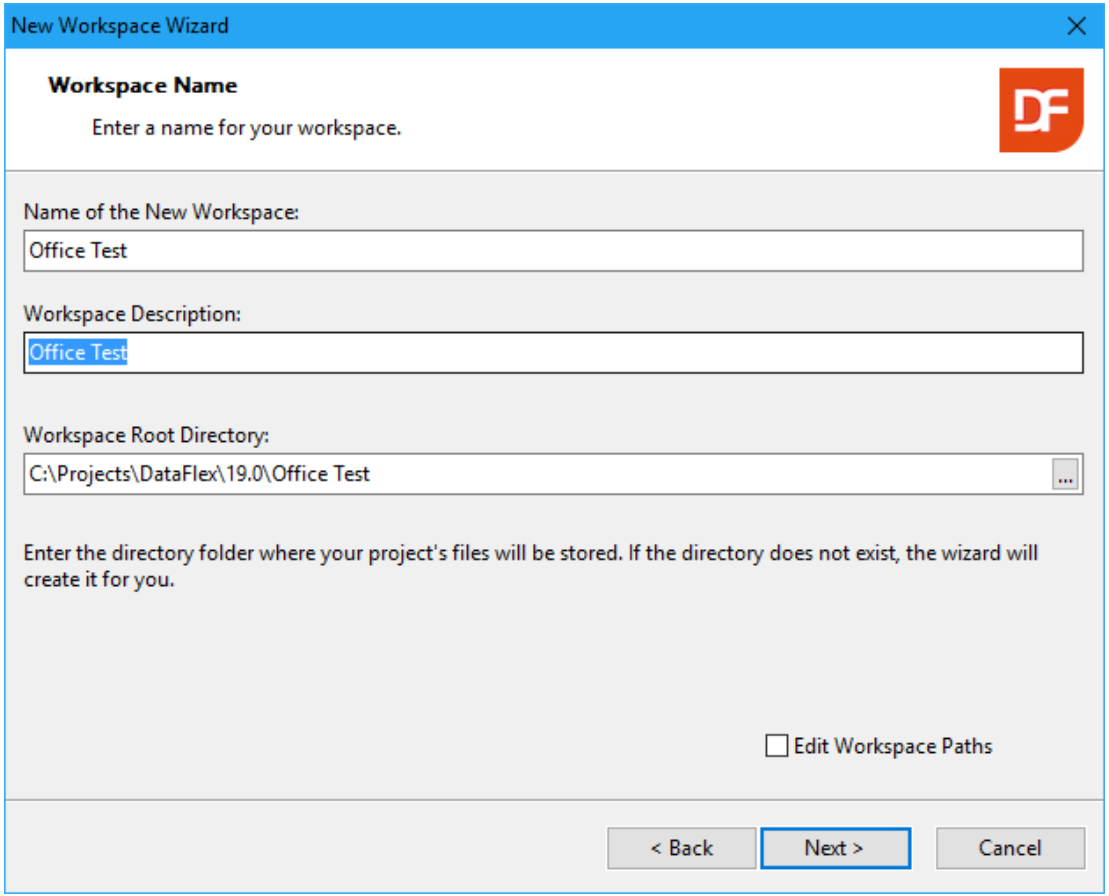
- Determine target version of the office product you will connect to
  - Newer versions have more, less or changed features
    - Using obsolete functions might not work
    - Using new functions will not work
  - Check version in Application object
    - Word, Excel & Outlook have a version number

Selecting the Office Version

- All versions of the office products use the same GUID
  - Against the rules
  - If there is an Office version installed the program connection can be made

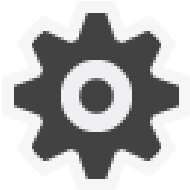
How to start?

- Create a test workspace
- Create a test Windows GUI application



How to start?

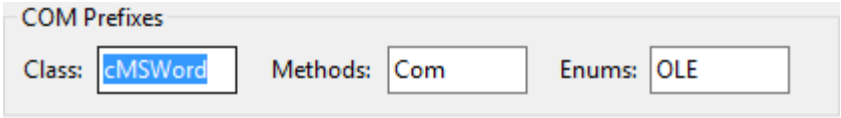
- Go to Create New | Class | Import COM Automation



- Search for the registered product you want to use or browse to the .OLB file
  - Excel & Word are present in the list
  - Outlook needs to go via the .OLB file

How to start?

- Use an application specific COM prefix



  - Word, Excel and Outlook all have an Application class
  - With default prefix (cCom) it would be impossible to combine two or more Office products in ONE DataFlex application due to class name clashes

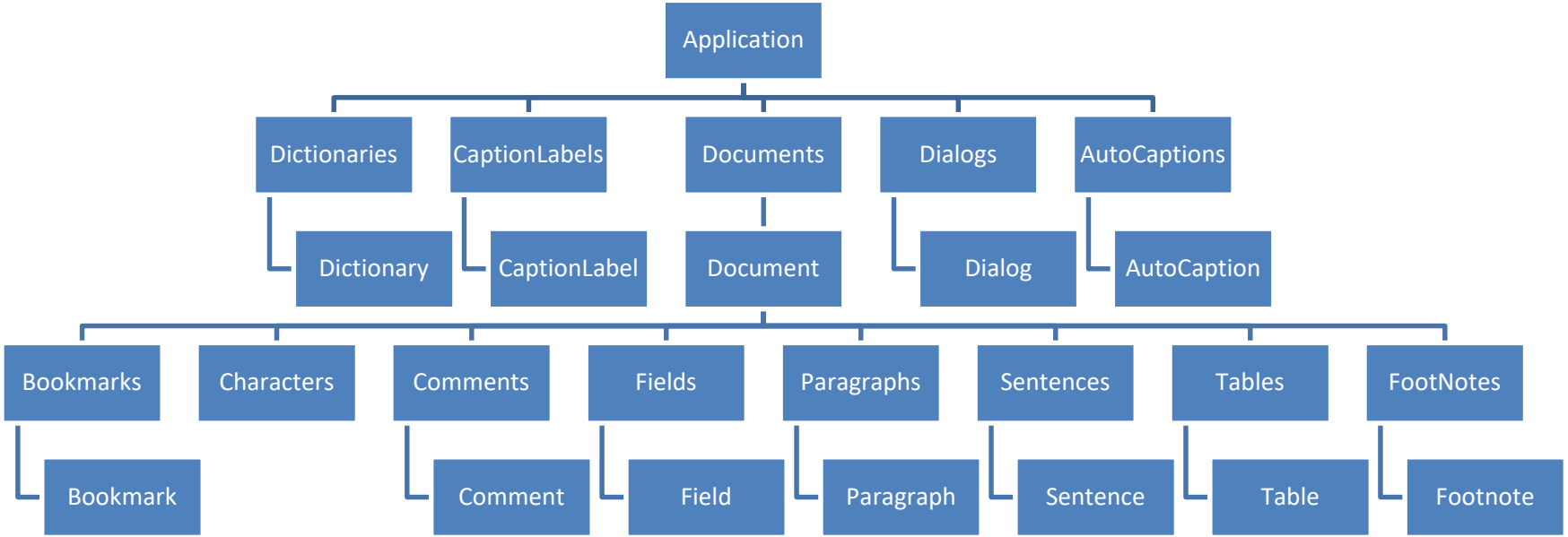
Using Microsoft Word via COM

How to attack…

Documentation (find it and read it!)

- Go to google and enter:
  msdn microsoft word vba
- Results in interesting topics:
  - VBA reference
  - Word Object Model Overview
  - Object model (Word VBA reference)

Word Object Model (partial)



Documentation

- What version of Word?
  - Customers might have an older version
    - Some features might not exist yet
    - Some features are reworked
  - So use the documentation that "matches" the version to use

Create Objects

- Create a DataFlex object for each of the Word objects that you want to address
  - No need to nest objects
  - Order of objects is not important
  - Multiple instances of the same class are allowed

```
Object oWordApplication is a cMSWordApplication
End_Object

Object oWordDocument is a cMSWordDocument
End_Object

Object oWordDocuments is a cMSWordDocuments
End_Object
```

Create Objects

- Write code to attach the DataFlex object to the Word counterpart
  - `Set pvComObject` to a Dispatch ID makes the connection
  - In most cases the Dispatch ID is retrieved via a function call
    - Some objects need to be created via `CreateComObject` or `AttachActiveObject`

Creating the Application Connection

- `CreateCOMObject` creates a new instance of Word
- `AttachActiveObject` attaches to an already running Word instance
  - Reduces resources but your application might get into trouble when Word is closed

Create Objects

- We suggest you create a new instance…

```
Send CreateComObject of oWordApplication
```

- Test if the object was created…

```
Get IsComObjectCreated of oWordApplication to bIsCreated
```

```
Get IsComObjectCreated of oWordApplication to bIsCreated
If (not (bIsCreated)) Begin
    Send CreateComObject of oWordApplication
    Get IsComObjectCreated of oWordApplication to bIsCreated
End
```

And Now?

- Get access to a document
  - Load an existing document
  - Create a new document
  - Enumerate the loaded documents

- So we look at the VBA doc and see…

Application Object Documentation

VBA

```
Set wrd = GetObject(, "Word.Application")
wrd.Visible = True
wrd.Documents.Open "C:\My Documents\Temp.doc"
Set wrd = Nothing
```

How to read this?

- `wrd` is a self defined reference to the word application object
- `Documents` is a property of the application object returning the dispatch ID of the Documents collection object
- `Open` is a method of the Documents object

VBA

```
Set wrd = GetObject(, "Word.Application")
wrd.Visible = True
wrd.Documents.Open "C:\My Documents\Temp.doc"
Set wrd = Nothing
```

Documentation

- How to translate?
- Think in an OO way
- Word can handle multiple documents in one instance, contains a collection object for handling the individual document objects
  - Objects usually have a property with the same name returning its object handle

Object Reference

- `Application.Documents` returns a Dispatch ID for the Documents collection object
  - It is like the handle variable in DataFlex containing an object ID

How to do this in DataFlex?

- First create the Documents object

```
Object oWordDocuments is a cMSWordDocuments
End_Object
```

- Get the result of the Documents property to a variant variable and assign that value to the Documents object

```
Get ComDocuments of oWordApplication to vDocuments
Set pvComObject of oWordDocuments to vDocuments
```

- Now you can access the methods of the Documents object

Use of Optional Parameters

Documents.Open Method (Word)

Syntax

expression .**Open**(***FileName***, ***ConfirmConversions***, ***ReadOnly***, ***AddToRecentFiles***, ***PasswordDocument***, ***PasswordTemplate***, ***Revert***, ***WritePasswordDocument***, ***WritePasswordTemplate***, ***Format***, ***Encoding***, ***Visible***, ***OpenConflictDocument***, ***OpenAndRepair***, ***DocumentDirection***, ***NoEncodingDialog***)

- In VB optional parameter can be skipped by using a comma or omitted if no other parameter is to be used
- In DataFlex optional parameters **MUST** be specified by using the NOTHING word

VB Open vs DataFlex Open

- VB:

```
wrd.Documents.Open "C:\My Documents\Temp.doc"
```

  - or

```
Documents.Open FileName:="C:\MyFiles\MyDoc.doc", ReadOnly:=True
```

- DataFlex:

```
Get ComOpen of oWordDocuments "c:\My Documents\Temp.doc" ;
NOTHING NOTHING NOTHING NOTHING NOTHING ;
NOTHING NOTHING NOTHING NOTHING NOTHING ;
NOTHING NOTHING NOTHING NOTHING NOTHING to vDocument
```

  - More typing but it works fine!

```
Send ComPrintOut of oWordDocument Nothing Nothing OLEwdPrintAllDocument ;
Nothing Nothing Nothing Nothing 1 Nothing OLEwdPrintAllPages ;
Nothing Nothing Nothing Nothing Nothing Nothing Nothing Nothing Nothing
```

Enumerate Collections

VBA

```
With Selection.Paragraphs
  .Alignment = wdAlignParagraphRight
  .LineSpacingRule = wdLineSpaceDouble
End With
```

- DataFlex does not have a direct `With..End With` support
  - Is that a problem? Can I not enumerate the objects in this collection?

Enumerate Collections in DataFlex

```
Get ComParagraphs of oWordDocument to vParagraphs
Set pvComObject of oDocumentParagraphs to vParagraphs

Get ComCount of oDocumentParagraphs to iParagraphs
For iParagraph from 1 to iParagraphs
    Get ComItem of oDocumentParagraphs iParagraph to vParagraph
    Set pvComObject of oDocumentParagraph to vParagraph

    Set ComFirstLineIndent of oDocumentParagraph to fPoints
Loop
```

- Above enumerates all paragraphs and indents each paragraph by 0.5 cm
  - `fPoints` value from `CentimetersToPoints`

Bookmarks Collection

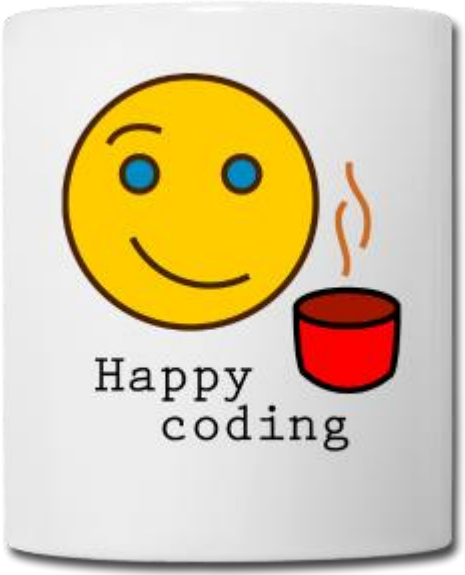
- Create a letter based on a template containing bookmarks for replacement
  - E.g. 1st bookmark for customer name, 2nd for customer address etc
- Enumerate these bookmarks from top-to-bottom (so N to 1)
  - Replacing a bookmark invalidates bookmarks with higher item references

Other Office products

Excel & Outlook

More of the Same

- Once you understand how the document structure works you can visualize the interface and doing more (in the same application or with others) will get easier
  - Practice, practice, practice…



What is needed to send an e-Mail?

```
Get IsComObjectCreated of oOutlookApplication to bIsCreated
If (not (bIsCreated)) Begin
    Send CreateComObject of oOutlookApplication
End

Get ComGetNamespace of oOutlookApplication "MAPI" to vNameSpace
Set pvComObject of oOutlookNameSpace to vNameSpace

Send ComLogon of oOutlookNameSpace "outlook" Nothing True False

Get ComGetDefaultFolder of oOutlookNameSpace OLEolFolderOutbox to vFolder
Set pvComObject of oOutlookFolder to vFolder

Get ComItems of oOutlookFolder to vItems
Set pvComObject of oOutlookItems to vItems

Get ComAdd of oOutlookItems OLEolMailItem to vMailItem
Set pvComObject of oOutlookMailItem to vMailItem
Set ComTo of oOutlookMailItem to sEmailAddress
Set ComSubject of oOutlookMailitem to "Overview of All your orders"
Set ComBody of oOutlookMailItem to "Find an overview of all your orders (past and current) in the PDF attachment"

Get ComAttachments of oOutlookMailItem to vAttachments
Set pvComObject of oOutlookAttachments to vAttachments

Get ComAdd of oOutlookAttachments sPDFFileName OLEolByValue Nothing 'Orders overview' to vAttachment

Send ComSend of oOutlookMailItem
```

Troubleshooting

Method Invocation Error

COM object method invocation error. Unknown error. Error code: 0x800706ba

Error: 4399

- Panic ?? No, the method executed has a problem. The error code tells what is wrong. Google for a solution:
- When you first launch Word programmatically, it connects to Word via an RPC server. When you close the document, this server gets closed without your application knowing about it.

  The solution is to trap an error and re-initialise your Word object. you will then be able to continue.

Method Invocation Error

COM object method invocation error. Attempt to use object before it's been initialized, pvComObject is NULL.

Error: 4399

- Oh oh…
- Simple solution:

```
Set pvComObject of oSomeObject to vSomeDispatch
```

Tips

Make a Precompile Package

- Add the class package to a self defined package so that it can be pre-compiled

header.pkg

```
01 Use DfAllEnt.pkg
02 Use MSWORD.pkg
```

Check if Word is Installed

```
Function IsWordInstalled Global Returns Boolean
    Boolean bInstalled
    String sClass

    Get CheckIfComRegistered "Word.Application" to sClass
    Move (sClass = "{000209FF-0000-0000-C000-000000000046}") to bInstalled

    Function_Return bInstalled
End_Function
```

- Technically it checks if the progID is present in the registry and not if the product is installed

Respond to a Quit event

- Closing the Office product causes wrong program statuses.
  - E.g. Word is closed, DataFlex object still thinks it is connected
- Use the `OnComQuit` event to release the connection

```
Object oWordApplication is a cMSWordApplication
    Procedure OnComQuit
        Send ReleaseComObject
    End_Procedure
```

LibXL

Excel Library

What is this?

- www.libxl.com says: Direct reading and writing Excel files
- Can be used without Excel being installed
- Usage everywhere but very useful when Excel should not be installed (e.g. on a webserver)

- Data Access Europe sells a library+DLL solution

That's it for today...