Windows-Automatisation 1

OLE-Automation/ActiveX-Automation, ooRexx Proxy Class "OLEObject"

Business Programming 1 | **Business Programming 2**

Basics, Parsing	Commands, APIs	Window-Automatisation, Web-Scripting	Security, Debugging	Graphical User Interfaces (GUI), Sockets, ...

Prof. Rony G. Flatscher

OLE (ActiveX) Automation, 1

- **COM**: **C**omponent **O**bject **M**odel
  - RPC ("remote procedure call")
  - Defines standard ***i****nterface* function "***I***Unknown"
  - Developed further
    - DCOM, COM+
- **OLE** : **O**bject **L**inking and **E**mbedding
  - COM-based, ***i****nterface* function "***I***Dispatch"
  - Linking of documents (dynamic data exchange, DDE)
    - Cold link
    - Warm link
    - Hot link
  - Embedding of alien/foreign documents

OLE (ActiveX) Automation, 2

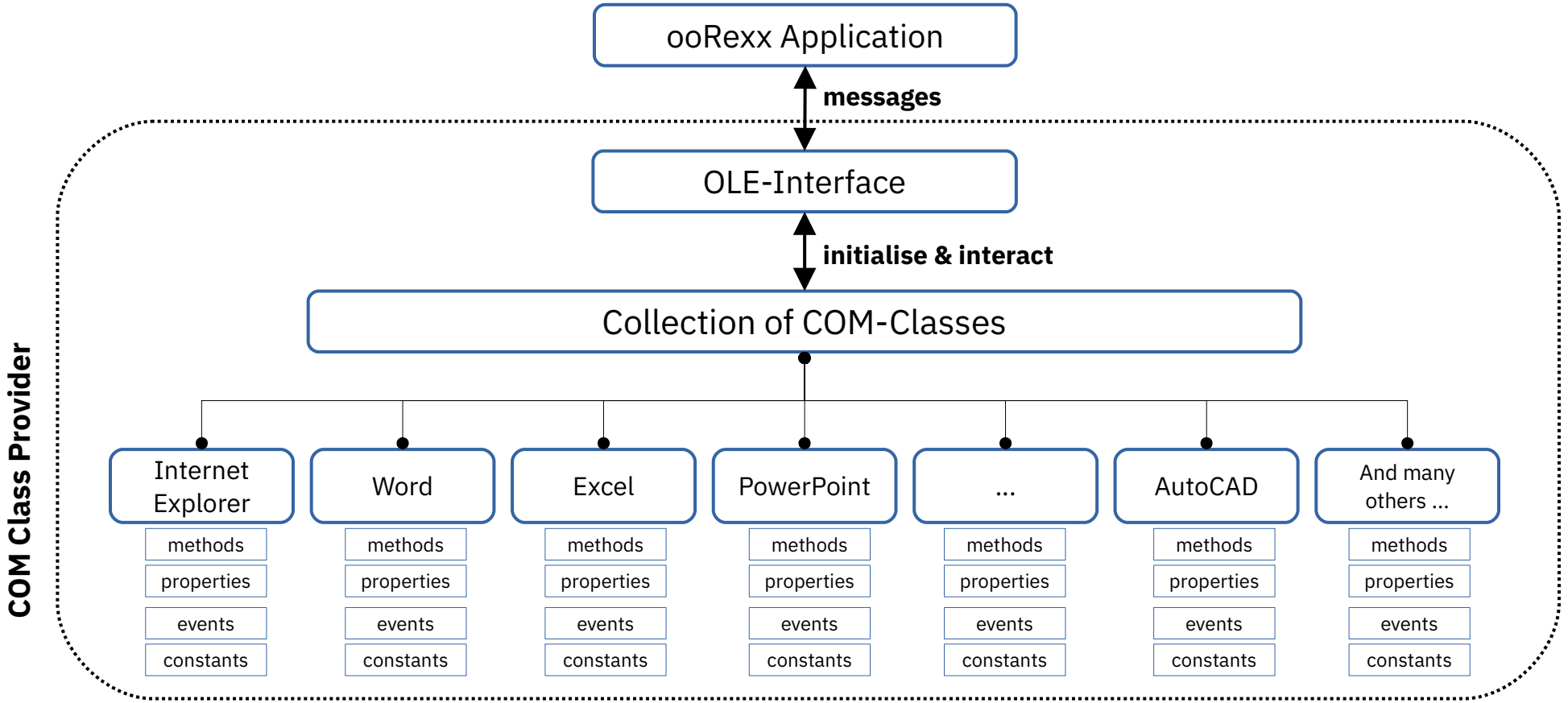
- VBX, OCX, ActiveX
  - Set of COM-Interfaces for defining Windows "components"
    - Windows programs, which can be combined as building stones
    - Defined interface for communicating with components
  - Acronyms
    - Visual Basic Extension
      - Mostly developed for GUI
    - Object Component Extension and ActiveX
      - Independent of Visual Basic, therefore usable for all Windows programs

OLE (ActiveX) Automation, 3

- OLE (ActiveX) Automation
  - Interface for remotely controlling Windows applications/components
  - Set of COM-based interface functions
    - Standardized definition of programming interfaces for (scripting) languages
      - Invocation of *methods* (functions) in a Windows program/component
      - Querying and setting *property* ("attribute") values of a Windows program/component
      - Intercepting of *events* that get raised by a Windows program/component
      - Querying of *constant* values in a Windows program/component
    - Logging of user actions, which can be transformed into a "script" ("macro") program

Cf. winextensions.pdf (8. OLE Automation)

OLE (ActiveX) Automation, 4



Cf. winextensions.pdf (8. OLE Automation)

OLE (ActiveX) Automation, 5

- Information about Windows applications and components is stored in the Windows Registry
  - HKEY_CLASSES_ROOT
    - CLSID
      - GUID resp. UUID
      - Global resp. Universal Unique Identifier
    - ProgID
      - A string that can be easily comprehended and memorized by humans
- Identifying (addressing) of a COM Windows component can be done via a *CLSID*, a *ProgID* or a "*moniker*" (a human readable string)

ooRexx Class ".OLEObject", 1

- Reference documentation available with the Windows version of ooRexx
  - "ooRexx Windows Reference": winextensions.pdf
- "Proxy" class for addressing OLE resp. ActiveX Windows programs, which allows for
  - Locating and addressing running OLE/ActiveX programs
  - Creating new instances of OLE/ActiveX programs
  - Querying the *published* programming interfaces (methods, properties, constants, events)
  - Sending ooRexx messages to invoke the interfaces
    - Arguments are automatically converted by ooRexx
    - Return values are automatically converted by ooRexx

Cf. winextensions.pdf (8.3. The OLEObject Class)

ooRexx Class ".OLEObject", 2

- Querying/setting of property values using ooRexx messages
  - Windows properties can be used as if they were ooRexx attributes
- Can intercept events and invoke ooRexx methods by the same name
- Automatically carries out the necessary datatype conversion between Windows and ooRexx, ie. between ooRexx and the following (COM) Windows datatypes
  - VT_EMPTY, VT_NULL, VT_VOID, VT_I1, VT_I2, VT_I4, VT_I8, VT_UI1, VT_UI2, VT_UI4, VT_UI8, VT_R4, VT_R8, VT_CY, VT_DATE, VT_BSTR, **VT_DISPATCH**, *VT_VARIANT*, **VT_PTR**, VT_SAFEARRAY

Cf. winextensions.pdf (8.3. The OLEObject Class)

Methods (1 of 2)

- Init(ProgID | CLSID [, NOEVENTS|WITHEVENTS])
  - Creates a new instance of the OLE/ActiveX program ("COM class")
  - Returns an ooRexx (proxy) object for it
- GetObject(Moniker [, SubclassOfOLEObject])
  - Class method, which searches an existing instance of a COM class
  - Returns an ooRexx (proxy) object for it
- GetConstant([someConstantName])
  - Returns the value for the constant named *someConstantName*
  - Returns all published constants with their defined values as a Rexx stem

Cf. winextensions.pdf (8.3. The OLEObject Class)

Methods (2 of 2)

- GetKnownEvents, GetKnownMethods
  - Returns a stem containing all published events and methods of the COM class
- GetOutParameters
  - Returns an array object with the "out" parameters of the last invocation
- Dispatch(MessageName [, Argument1, Argument2, ...])
  - Invokes a method on the Windows side which has the same name as "MessageName" and forwards any supplied arguments
- UNKNOWN(MessageName [, ArrayWithArguments])
  - This method processes all unknown messages and forwards them to the OLE/ActiveX program, which gets represented by the proxy object
  - The reason why one can send ooRexx messages successfully to Windows objects!

Cf. winextensions.pdf (8.3. The OLEObject Class)

Hint about the examples

- The Windows version of Open ooRexx (ooRexx) is delivered with numerous OLE/ActiveX examples, which can be found under the ooRexx installation directory:

  ```
  ?\ooRexx\samples\ole
  ```

- Attention!
  - The following foils only depict a subset of the supplied examples
  - Therefore please study and run all examples in all of the samples subdirectories
    - These examples do not change your computer settings permanently!

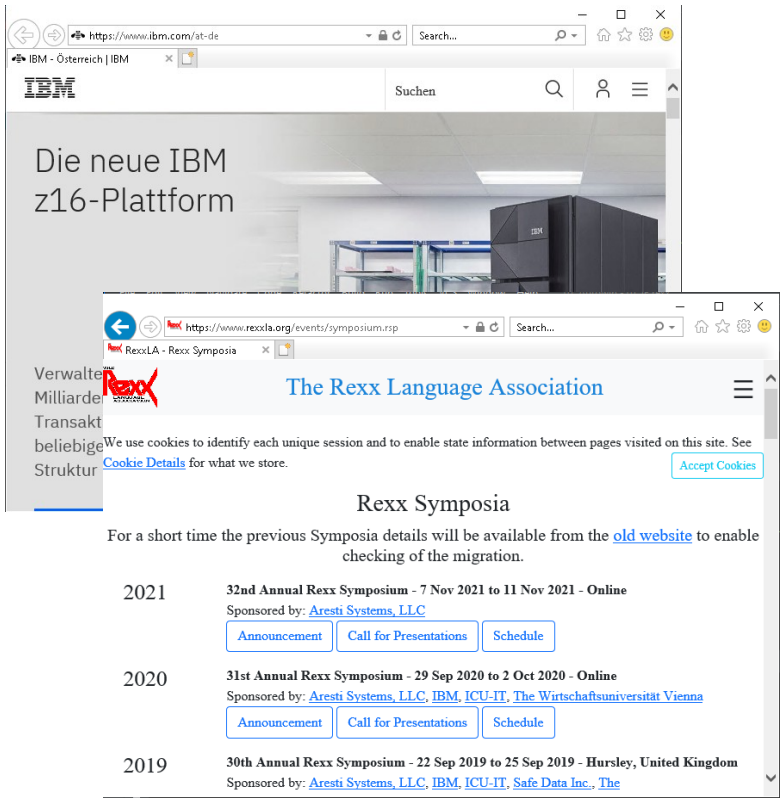
"InternetExplorer.Application" # 1

- ...\ole\apps\MSInternetExplorer_navigate.rex (executed on 2022-05-03)

```
/* create an object for IE */
myIE = .OLEObject~New("InternetExplorer.Application")

myIE~Width  = 1000
myIE~Height = 800
Say "current dimensions of IE are:" myIE~Width "by" myIE~Height

/* set new dimensions and browse IBM homepage */
myIE~Width  = 1280
myIE~Height = 1024
myIE~Visible = .True    -- now show the window
myIE~Navigate("https://www.ibm.com")

/* wait for 10 seconds */
Call SysSleep 10

/* wait for 10 seconds */
myIE~Navigate("https://www.rexxla.org/events")
Call SysSleep 10
myIE~quit
```

Output:

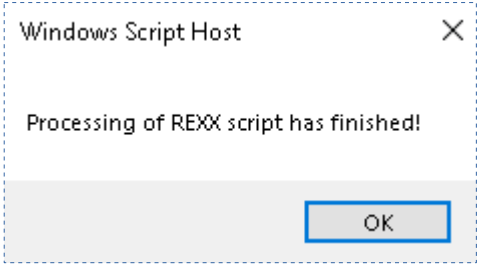
```
current dimensions of IE are: 1000 by 800
```



"WScript.Shell"

- ...\ole\apps\WScriptShell.rex

```
WshShellObj = .OLEObject~New("WScript.Shell")

WshEnv = WshShellObj~Environment
Say "Operating system:" WshEnv["OS"]
Say "You have" WshEnv["NUMBER_OF_PROCESSORS"] "processor(s) of",
    WshEnv["PROCESSOR_ARCHITECTURE"] "architecture in your system."

Say "The following directories represent special folders on your system:"
Do Folder Over WshShellObj~SpecialFolders
  Say "    " Folder
End

Say "Creating a shortcut for NOTEPAD.EXE on your Desktop..."
Desktop = WshShellObj~SpecialFolders("Desktop")
ShortCut = WshShellObj~CreateShortcut(Desktop || "\Shortcut to Notepad.lnk")
ShortCut~TargetPath = "%WINDIR%\notepad.exe"
ShortCut~Save

WshShellObj~Popup("Processing of REXX script has finished!")
```



Possible Output:

```
Operating system: Windows_NT
You have 2 processor(s) of AMD64 architecture in your system.
The following directories represent special folders on your system:
    C:\Users\Public\Desktop
    C:\ProgramData\Microsoft\Windows\Start Menu
    … cut …

Creating a shortcut for NOTEPAD.EXE on your Desktop...
```

"WScript.Network"

- ...\ole\apps\WScriptNetwork.rex

```
WshNetObj = .OLEObject~New("WScript.Network")

Say "Computer Name:" WshNetObj~ComputerName
Say "User Domain:" WshNetObj~UserDomain
Say "User Name:" WshNetObj~UserName

Say "The following network drives are currently mapped:"
MappedDrives = WshNetObj~EnumNetworkDrives
Do i=0 To MappedDrives~Count/2 - 1
  Say "   Drive" MappedDrives[i*2] "is mapped to" MappedDrives[i*2 + 1]
End

Say "The following network printers are currently connected:"
Printers = WshNetObj~EnumPrinterConnections
Do i=0 To Printers~Count/2 - 1
  Say "   Port" Printers[i*2] "is connected to" Printers[i*2 + 1]
End
```

```
MappedDrives:
[0] Z:
[1] \\192.168.122.1\Shared

Printers:
[0] PORTPROMPT:
[1] Microsoft Print to PDF
[2] SHRFAX:
[3] Fax
```

Possible Output:

```
Computer Name: DESKTOP-NS1T6G3
User Domain: DESKTOP-NS1T6G3
User Name: waldi
The following network drives are currently mapped:
   Drive Z: is mapped to \\192.168.122.1\Shared
The following network printers are currently connected:
   Port PORTPROMPT: is connected to Microsoft Print to PDF
   Port SHRFAX: is connected to Fax
```

"Excel.Application" # 1 (1 of 3)

- ...\ole\apps\MSExcel.rex

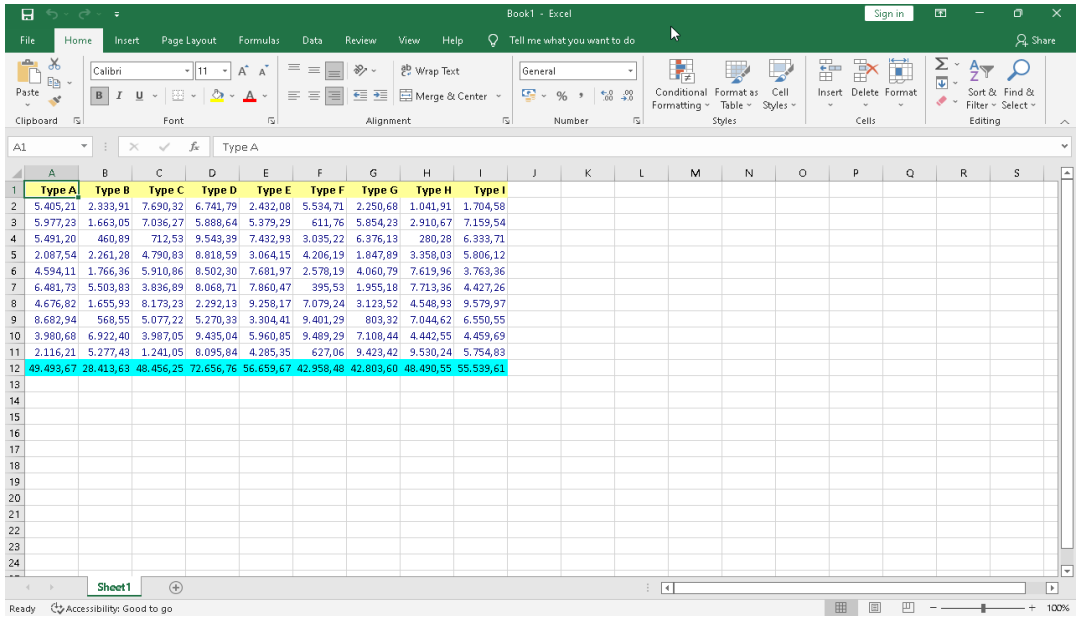
```
excelApplication = .OLEObject~new("Excel.Application")
excelApplication~visible = .true             -- make Excel visible
Worksheet = excelApplication~Workbooks~Add~Worksheets[1]
colTitles = "ABCDEFGHI"                      -- define first nine column letters
lastLine = 12                                -- number of lines to process
sumFormula = "=sum(?2:?"lastLine-1")"        -- English formula: question marks will be changed to column letter
say "sumFormula:      " sumFormula "(question marks will be changed to column letter)"
xlHAlignRight = excelApplication~getConstant("xlHAlignRight") -- get value of "horizontal align right" constant
do line = 1 to lastLine                      -- iterate over lines
  do col = 1 to colTitles~length             -- iterate over columns
    colLetter = colTitles[col]               -- get column letter
    cell = Worksheet~Range(colLetter||line)  -- e.g. ~Range("A1")
    if line = 1 then do                 -- first row? yes, build title
      cell~value = "Type" colLetter          -- header in first row
      cell~font~bold = .true                 -- make font bold
      cell~Interior~ColorIndex = 36          -- light yellow
      cell~style~horizontalAlignment = xlHAlignRight  -- right adjust title
    end
    else if line = lastLine then do     -- last row? yes, build sums
      /* set formula, e.g. "=sum(B2:B9)" */
      cell~formula = sumFormula~changeStr("?",colLetter) -- adjust formula to column to sum up
      cell~Interior~ColorIndex = 8           -- light blue
    end
    else do -- a row between 2 and 9: fill with random values
      cell~value = random(999999) / 100      -- create a random decimal value
      cell~font~ColorIndex = 11              -- set from black to violet
    end
  end
end
```

"Excel.Application" # 1 (2 of 3)

- ...\ole\apps\MSExcel.rex (continued)

```
sumCell = WorkSheet~range("A"lastLine)       -- get sum-cell of column A
if sumCell~text = "#NAME?" then
do
   say
   say "** Excel reports a '#NAME?' error for the 'sum' function! Probable cause: **"
   say "** your local Excel user interface language is not set to English, therefore you need **"
   say "** to adjust the function name 'sum' in the variable 'sumFormula' to your user interface **"
   say "** language and rerun this program (e.g. in German you need to rename 'sum' to 'summe') **"
   say "** sumCell~formula:" sumCell~formula
   say "** sumCell~text:   " sumCell~text
   say "** sumCell~value:  " sumCell~value
   say
end
formatString = "#"excelApplication~thousandsSeparator"##0"excelApplication~decimalSeparator"00"
say "formatString:    " formatString           -- show format string
excelApplication~useSystemSeparators = .false   -- allow our format string to be used everywhere
stringRange="A2:"colTitles~right(1)lastLine
say "formatting range:" stringRange
WorkSheet~range(stringRange)~numberFormat = formatString -- get range and set its number format
excelApplication~DisplayAlerts = .false      -- no alerts from now on
homeDir = value("USERPROFILE",,"ENVIRONMENT")-- get value for environment variable "USERPROFILE"
fileName = homeDir"\samp09_ooRexx.xlsx"      -- build fully qualified filename
say "fully qualified fileName:" fileName     -- show fully qualifed filename
Worksheet~SaveAs(fileName)                   -- save file
   -- let the user inspect the Excel file
say "Excel sheet got saved to file, press enter to continue ..."
parse pull .                                 -- wait for user to press enter
excelApplication~Quit                        -- close Excel
```

"Excel.Application" # 1 (3 of 3)



Possible Output:

```
sumFormula:       =sum(?2:?11) (question marks will be changed to column letter)
formatString:     #.##0,00
formatting range: A2:I12
fully qualified fileName: C:\Users\waldi\samp09_ooRexx.xlsx
Excel sheet got saved to file, press enter to continue …
```

"Scripting.FileSystemObject"

- ...\ole\apps\ScriptingFileSystemObject.rex

```
fsObject = .OLEObject~new("Scripting.FileSystemObject")
allDrives = fsObject~drives
if allDrives = .NIL then do
  say "The object did not return information on your drives!"
  exit 1
end

do i over allDrives
  info = i~DriveLetter "-"
     /* show the DriveType in human-readable form */
  j = i~DriveType
  select
    when j=1 then info = info "Removable"
    when j=2 then info = info "Fixed"
    when j=3 then info = info "Network"
    when j=4 then info = info "CD-ROM"
    when j=5 then info = info "RAM Disk"
    otherwise info = info "Unknown"
  end

     /* append the ShareName for a network drive... */
  if j=3 then info = info i~ShareName
     /* ...and the VolumeName for the other ones */
  else if i~IsReady then info = info i~VolumeName
  say info
end
```

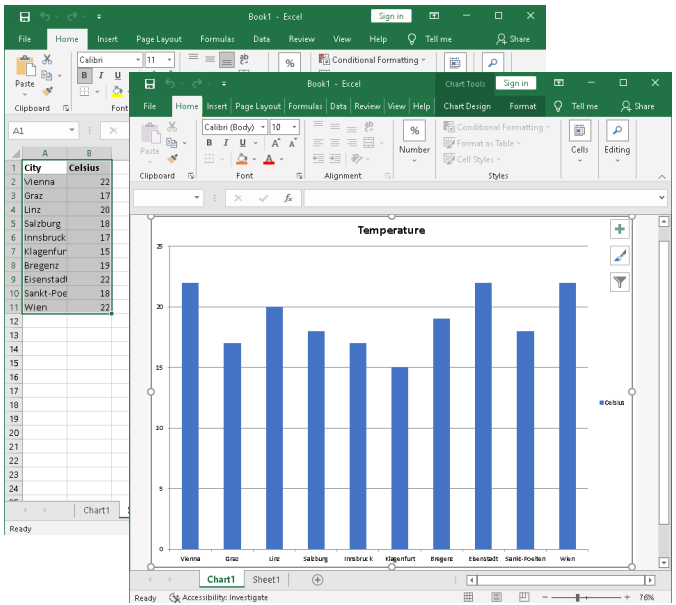
Possible Output:

```
C - Fixed
Z - Network \\192.168.122.1\Shared
```

"Excel.Application" # 2

...\ole\apps\MSExcel_cURL.rex

```
-- Get information using curl, save results in array
cityArr = .array~of("Vienna", "Graz", "Linz", "Salzburg", "Innsbruck", "Klagenfurt", "Bregenz", -
                    "Eisenstadt", "Sankt-Poelten", "Wien")
cityWeather = .array~new                                    -- array for weather information
do counter i name over cityArr
    command='curl https://wttr.in/'name'?format="%l:+%t"'   -- https://github.com/chubin/wttr.in
    say "#" i":" command                                    -- give user feedback
    outArr = .array~new                                     -- array for stdout
    ADDRESS SYSTEM command with output using (outArr) error using (.array~new)
    cityWeather~append(outArr[1])                           -- append output to array
end

-- Start Excel with empty worksheet
excelApplication = .OLEObject~new("Excel.Application")
excelApplication~visible = .true                            -- make Excel visible
Worksheet = excelApplication~Workbooks~Add~Worksheets[1]    -- add worksheet
-- Create bold collum header in first row
colhead = .array~of("City", "Celsius")                      -- array with header text
do counter col name over colhead
    Worksheet~cells(1,col)~Value = name                     -- insert items from array
    Worksheet~cells(1,col)~font~bold = .true                -- make bold
end
-- Insert information from gained with curl
do counter row name over cityWeather
    parse var name city ":" temperature "°C"               -- parse curled informations
    Worksheet~cells(row+1,1)~Value = city                   -- city in collum 1
    Worksheet~cells(row+1,2)~Value = temperature            -- temperature in collum 2
end
-- Select range for chart, use left upper and lower right cell location
Worksheet~Range("A1:B" || (row+1))~Select                   -- include title row
-- Add Chart
excelApplication~Charts~Add                                 -- create new chart
excelApplication~ActiveChart~HasTitle = .True               -- add title
excelApplication~ActiveChart~ChartTitle~Characters~Text = "Temperature"
```



Output:

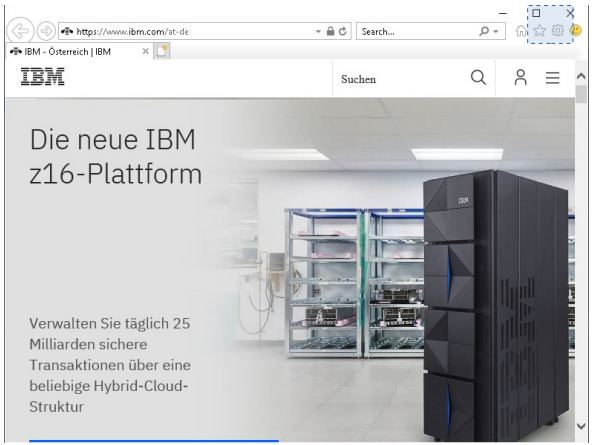
```
# 1: curl https://wttr.in/Vienna?format="%l:+%t"
# 2: curl https://wttr.in/Graz?format="%l:+%t"
# 3: curl https://wttr.in/Linz?format="%l:+%t"
# 4: curl https://wttr.in/Salzburg?format="%l:+%t"
# 5: curl https://wttr.in/Innsbruck?format="%l:+%t"
# 6: curl https://wttr.in/Klagenfurt?format="%l:+%t"
… cut …
```


Windows 10 Version 21H1, Excel 2016 Version 2203

"InternetExplorer.Application" # 2

- ...\ole\apps\ MSInternetExplorer_events.rex

```
/* instantiate an instance of the Internet Explorer */
myIE = .watchedIE~new("InternetExplorer.Application","WITHEVENTS")
myIE~visible = .true
myIE~navigate("http://www.ibm.com/")

/* wait for the OnQuit event of the browser to change */
/* the !active attribute of the REXX object to false  */
myIE~!active = .true
do while myIE~!active = .true
  call syssleep(2)
end

::CLASS watchedIE SUBCLASS OLEObject
/* ... Cut ... Lines deleted, please lookup the original file in your installation ! */
/* this is an event of the Internet Explorer */
::METHOD TitleChange
  use arg Text
  say "The title has changed to:" text
/* this is an event of the Internet Explorer */
::METHOD OnQuit
  self~!active = .false    -- terminates the waiting loop in main code

::METHOD !active ATTRIBUTE -- store the active attribute
```



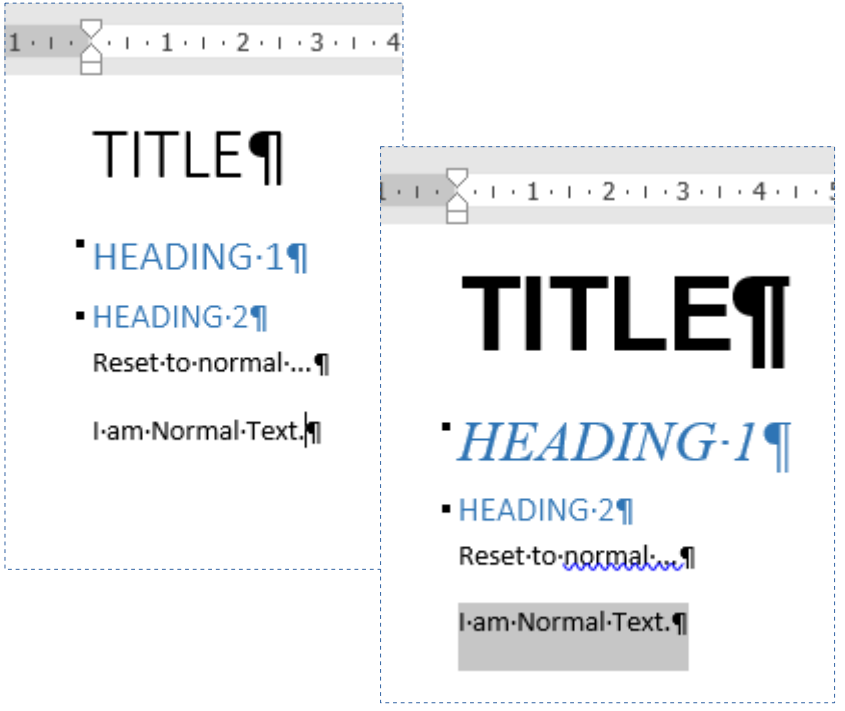
Output:

```
The title has changed to: IBM - Österreich | IBM
```

"Word.Application"

...\ole\apps\MSWord_useStyles.rex

```
Word = .OLEObject~New("Word.Application")
Word~Visible = .TRUE                        -- make Word visible
Document = Word~Documents~Add               -- add document
Selection = Word~Selection                  -- selection object
say "# 1: Create: title style..."
Selection~Style = "Title"                   -- Create selection with style: Title
Selection~TypeText("TITLE")                 -- give selection a text
Selection~TypeParagraph                     -- add paragraph (go to next line)
say "# 2: Create: heading 1 style..."
Selection~Style = "Heading 1"               -- Create selection with style: Heading 1
Selection~TypeText("HEADING 1")             -- give selection a text
Selection~TypeParagraph                     -- add paragraph (go to next line)
say "# 3: Create: heading 2 style..."
Selection~Style = "Heading 2"               -- Create selection with style: Heading 2
Selection~TypeText("HEADING 2")             -- give selection a text
Selection~TypeParagraph                     -- add paragraph (go to next line)
   -- Note: Usually the style "Normal" follows heading styles
Selection~TypeText("Reset to normal ...") -- "Normal" follows "Heading 2"
Selection~TypeParagraph                     -- add paragraph (go to next line)
say "# 4: Create: normal text style..."
Selection~Style = "Normal"                  -- Create selection with style: normal
Selection~TypeText("I am Normal Text.")     -- give selection a text

say "Press any key to change style!"
parse pull                                  -- wait for key press

-- Go through each sentence
SentenceCount = Document~Sentences~Count        -- get sentence count
Font = Selection~Font                           -- get font object
do SentenceNumber = 1 to SentenceCount          -- go through each sentence

   Document~Sentences(SentenceNumber)~Select    -- select sentence
   StyleName = Selection~Style~NameLocal        -- get style name of sentence
   Select case StyleName                        -- make changes depending on style name
      when "Title" then do
          Font~Name="Arial"
          Font~Size="38"
          Font~Bold = .TRUE
      end
      when "Heading 1" then do
          Font~Name="Times New Roman"
          Font~Size="22"
          Font~Italic = .TRUE
      end
      otherwise NOP
   end
end
```



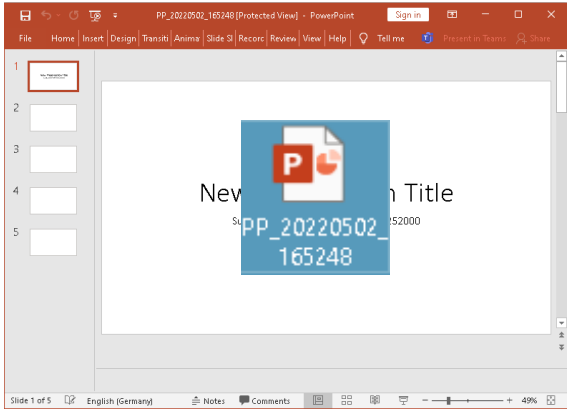
Output:

```
Press any key to change style!
```

"PowerPoint.Application"

...\ole\apps\MSPowerPoint_layouts.rex

```
-- Start PowerPoint
PP = .OLEObject~New("PowerPoint.Application")
PP~Visible = .TRUE                                          -- make PP visible
-- Add first Slide
PP~Presentations~Add~Slides~Add(1, 1)                       -- add: Title-Slide
FirstSlide = PP~ActivePresentation~Slides(1)
FirstSlide~Shapes(1)~TextFrame~TextRange = "New Presentation Title"   -- first Shape is Title
FirstSlide~Shapes(2)~TextFrame~TextRange =  "Subtitle at:" .dateTime~new -- second Shape is SubTitle
-- Append several Slides
ActivePresentation = PP~ActivePresentation
ActivePresentation~Slides~Add(2, 1)                         -- append: second Title-Slide
ActivePresentation~Slides~Add(3, 2)                         -- append: Title and Text Slide
ActivePresentation~Slides~Add(4, 3)                         -- append: Title and 2-Columne Text
ActivePresentation~Slides~Add(5, 5)                         -- append: Title, Text and Chart
-- Check what on Slide
SlideCount = ActivePresentation~Slides~Range~Count          -- get amount of Slides
say "Your Presentation has" SlideCount "Slides"
do SlideNumber = 1 to SlideCount                            -- go through each Slide
    ShapeCount = ActivePresentation~Slides(SlideNumber)~Shapes~Range~Count  -- get amount of Shapes per Slide
    say " Slide" SlideNumber "has" ShapeCount "Shapes"
    do ShapeNumber = 1 to ShapeCount                                -- go through each Shape
        -- List names of Shapes on Slide
        say "  ("ShapeNumber"/"ShapeCount"):" ActivePresentation~Slides(SlideNumber)~Shapes(ShapeNumber)~name
    end
end
-- Save and Close
parse source . . s          -- get fully qualified path of this script
-- create unique file name
fileOut = filespec('location',s)"\PP_" || date("S") || "_" || changeStr(":",time(),"") || ".pptx"
ActivePresentation~SaveAs(fileOut)                          -- save pptx file
PP~Quit                                                     -- close PP
```



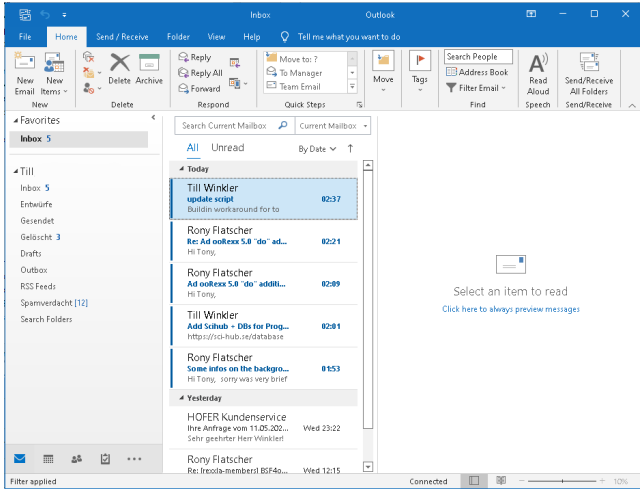
Possible Output:

```
Your Presentation has 5 Slides
 Slide 1 has 2 Shapes
  (1/2): Title 1
  (2/2): Subtitle 2
 Slide 2 has 2 Shapes
  (1/2): Title 1
  (2/2): Subtitle 2
 Slide 3 has 2 Shapes
  (1/2): Title 1
  … cut …
```

"Outlook.Application"

...\ole\apps\MSOutlook.rex

```
Outlook = .OLEObject~new("Outlook.Application")
NameSpace = Outlook~GetNamespace("MAPI")

-- In Outlook folder have numbers:
-- DeletedItems (3), Outbox (4), SentMail (5), Inbox (6),
-- Calendar (9), Contacts (10), Journal (11), Notes (12),
-- Tasks (13)
Inbox = NameSpace~GetDefaultFolder("6")      -- selects Inbox

-- makes Outlook visible and shows Inbox
Inbox~Display

InboxItems = Inbox~Items
MailCount = InboxItems~Count                 -- count items in Inbox
say "You have" MailCount "Mail(s) in your Inbox:"

Do ItemNumber = 1 to MailCount               -- go through each item
  Item = Inbox~Items(ItemNumber)
  Sender = Item~Sender~Name                  -- sender of item
  say "#" ItemNumber"." Sender
end
```



Possible Output:

```
You have 8 Mail(s) in your Inbox:
# 1. Rony Flatscher
# 2. Rony Flatscher
# 3. Till Winkler
# 4. Rony Flatscher
… cut …
```

"Access.Application"

- ...\ole\apps\MSAccess_contacts.rex

```
-- Initialize string to database path.
strDB = "c:\temp\newdb.mdb"
-- Create new instance of Microsoft Access.
appAccess = .OLEObject~new("Access.Application")
-- Open database in Microsoft Access window.
appAccess~NewCurrentDatabase("strDB")
-- Get Database object variable.
dbs = appAccess~CurrentDb
-- Create new table.
tdf = dbs~CreateTableDef("Contacts")
-- Create field in new table.
/* Please note how to access the constant.
   Microsoft documentation and the MS OLEViewer output
   these constants as dbText, dbBinary, etc. - the type library
   however prints them as DB_TEXT, DB_BINARY, etc.. Unless
   documentation is found why the names should be translated,
   the OLE code will *NOT* convert the names. */
fld = tdf~CreateField("CompanyName", appAccess~getConstant("db_Text"), 40)
-- Append Field and TableDef objects.
tdf~Fields~Append(fld)
dbs~TableDefs~Append(tdf)

appAccess~quit
```

ADS – Active Directory Services

- Authentication and authorization of users and computers
  - Users
  - Access rights
  - …
- Wikipedia overview article
  - https://en.wikipedia.org/wiki/Active_Directory

Collecting Information Using ADS, 1

Retrieve information about a computer with ADSI.

- ...\ole\adsi\adsi1.rex

```
ComputerName = value("COMPUTERNAME",,"ENVIRONMENT")
myComputer = .OLEObject~GetObject("WinNT://"||ComputerName||",computer")

say "Standard properties of this computer:"
say left("Name:"   ,10) myComputer~name
say left("Class:"  ,10) myComputer~class
say left("GUID:"   ,10) myComputer~guid
say left("ADsPath:",10) myComputer~adspath
say left("Parent:" ,10) myComputer~parent
say left("Schema:" ,10) myComputer~schema
```

Possible Output:

```
Standard properties of this computer:
Name:      DESKTOP-NS1T6G3
Class:     Computer
GUID:      {DA438DC0-1E71-11CF-B1F3-02608C9E7553}
ADsPath:   WinNT://WORKGROUP/DESKTOP-NS1T6G3
Parent:    WinNT://WORKGROUP
Schema:    WinNT://WORKGROUP/Schema/Computer
```

Collecting Information Using ADS, 2

Get a user's full name and change it..

- ...\ole\adsi\adsi2.rex
  (Note: this program requires administrator permissions)

```
ComputerName = value("COMPUTERNAME",,"ENVIRONMENT")      -- get ComputerName
UserID       = value("USERNAME",,"ENVIRONMENT" )         -- get UserName

userObject = .OLEObject~GetObject("WinNT://"||ComputerName||"/"||UserID||",user")

/* using the object property */
say "The full name for" UserID "is" userObject~FullName

/* using the standard get method for ADSI objects */
say "The full name for" UserID "is" userObject~Get("FullName")

say "Would you like to rename the full name (y/n)?"
pull answer

if answer = "Y" then do
  say "New full name:"
  parse pull answer

  /* set the property                                                */
  /* as an alternative, the property can also be set with the standard put */
  /* method of ADSI objects:                                         */
  /* userObject~Put("FullName",answer)                               */
  userObject~FullName=answer

  /* because properties are cached to avoid network calls, changing the    */
  /* properties of an object will only affect the cache at first.          */
  /* the object gets updated with the SetInfo method:                      */
  userObject~SetInfo

  say "updated the full name for" UserID
end
```

Possible Output:

```
rexx adsi2.rex
The full name for Administrator is John Doe
The full name for Administrator is John Doe
Would you like to rename the full name (y/n)?
y
New full name:
John Doe's Mother
updated the full name for Administrator
```

Collecting Information Using ADS, 3

Displaying namespaces and domains.

- ...\ole\adsi\adsi5.rex

```
myADS = .OLEObject~GetObject("ADs:")

do namespace over myADS
  say "Domains in" namespace~Name
  do domain over namespace
    if domain \= .nil then
      say "    " domain~name
    else
      say domain
  end
end
```

Possible Output (on a standalone Windows PC):

```
Domains in WinNT:
Domains in LDAP:
```

Collecting Information Using ADS, 4

Inspecting properties of an object.

- ...\ole\adsi\adsi6.rex

```
ComputerName = value("COMPUTERNAME",,"ENVIRONMENT")

myDomain = .OLEObject~GetObject("WinNT://"||ComputerName)
mySchemaClass = .OLEObject~GetObject(myDomain~schema)

say "Properties for the" myDomain~name "object:"
say

if mySchemaClass~container = 1 then do
  say myDomain~name "may contain the following objects:"
  do i over mySchemaClass~Containment
    say "    " i
  end
end
else
  say myDomain~name "is not a container."

say
say "Mandatory properties:"
do i over mySchemaClass~MandatoryProperties
  say "    " i
end

say
say "Optional properties:"
do i over mySchemaClass~OptionalProperties
  say "    " i
End
```

Possible Output:

```
Properties for the DESKTOP-NS1T6G3 object:

DESKTOP-NS1T6G3 may contain the following
objects:
    User
    Group
    Service
    FileService
    PrintQueue

Mandatory properties:

Optional properties:
    Owner
    Division
    OperatingSystem
    OperatingSystemVersion
    Processor
    ProcessorCount
    Name
```

Collecting Information Using ADS, 5

Create a group and several users in it.

- ...\ole\adsi\adsi7.rex

  (Note: this program requires administrator permissions)

```
ComputerName = value("COMPUTERNAME",,"ENVIRONMENT")  -- get ComputerName
computer = .OLEObject~GetObject("WinNT://"||ComputerName)
/* create a new group */
newGroup = computer~Create("group", "REXX-TestGroup")
newGroup~Description = "A test group created with REXX"
newGroup~SetInfo

/* make sure the information in the object cache is up-to-date */
newGroup~GetInfo
say "Created new group" newGroup~Name
say "Description:" newGroup~Description; say
say "Creating 15 users in this group:"
say "User01..User15 with passwords demo01..demo15"
/* create several new users */
do i = 1 to 15
  /* create name and other information */
  userName = "User"right(i,2,'0')
  userFullName = "Demo User Number" i
  userDescription = "A demo user that was created with REXX"
  userPassword = "demo"right(i,2,'0')

  newUser = computer~Create("user", userName)
  newUser~FullName = userFullName
  newUser~Description = userDescription
  newUser~SetPassword(userPassword)
  newUser~SetInfo
  newGroup~Add(newUser~ADsPath)
end
```

Output:

```
Created new group REXX-TestGroup
Description: A test group created with REXX

Creating 15 users in this group:
User01..User15 with passwords demo01..demo15
```

Collecting Information Using ADS, 6

Remove the users and the group that were created in *adsi7.rex*.

- ...\ole\adsi\adsi8.rex
  (Note: this program requires administrator permissions)

```
ComputerName = value("COMPUTERNAME",,"ENVIRONMENT")  -- get ComputerName
computer = .OLEObject~GetObject("WinNT://"||ComputerName)

say "Removing the fifteen users..."
do i = 1 to 15
  computer~Delete("user", "User"||right(i,2,'0'))
end

say "Removing the test group..."

computer~Delete("group", "REXX-TestGroup")

say "done"
```

Possible Output:

```
Removing the fifteen users...
Removing the test group...
```

WMI – Windows Management Instrumentation

- Allows to get information of instrumented components and notifications
  - Local and remote computers and devices
  - …
- Wikipedia overview article
  - https://en.wikipedia.org/wiki/Windows_Management_Instrumentation

Collecting Information Using WMI, 1

- ...\ole\wmi\accounts.rex

```
WMIObject = .OLEObject~GetObject("WinMgmts:{impersonationLevel=impersonate}")
userAccounts = WMIObject~InstancesOf("Win32_Account")

do instance over userAccounts
  say
  say "="~copies(16) instance~name "="~copies(16)
  do i over instance~properties_
    say left(i~name":",20,' ') i~value
  end
end
```

Possible Output:

```
================ Everyone ================
Caption:             DESKTOP-NS1T6G3\Everyone
Description:         DESKTOP-NS1T6G3\Everyone
Domain:              DESKTOP-NS1T6G3
InstallDate:         The NIL object
LocalAccount:        1
Name:                Everyone
SID:                 S-1-1-0
SIDType:             5
Status:              OK

================ LOCAL ================
Caption:             DESKTOP-NS1T6G3\LOCAL
… cut …
```

Collecting Information Using WMI, 2

- ...\ole\wmi\process.rex

```
WMIObject = .OLEObject~GetObject("winmgmts:{impersonationLevel=impersonate}")

say "Here is a list of currently running processes"

do process over WMIObject~InstancesOf("win32_process")
  say process~processid process~name
end
```

Possible Output:

```
Here is a list of currently running processes
0 System Idle Process
4 System
92 Registry
328 smss.exe
436 csrss.exe
508 wininit.exe
516 csrss.exe
600 services.exe
608 winlogon.exe
620 lsass.exe
752 svchost.exe
…cut…
```

Further Links, 1

- Rexx Language Association (RexxLA)
  - Numerous additional links

    <http://www.RexxLA.org/> (2022-04-22)
- OLE-/Active-X Query Tool written in ooRexx
  - ooRexx 5.1.0 (Windows): `oorexx\samples\ole\oleinfo` or for earlier versions

    <https://wi.wu.ac.at/rgf/wu/lehre/autowin/material/resources/oleinfo.zip> (2024-08-07)
- ooRexx related OLE/ActiveX page, a ***must*** to visit!

  <https://wi.wu.ac.at/rgf/wu/lehre/autowin/material/resources/pragmaticlee_archive_edited.zip> (2024-08-07)

Further Links, 2

- Microsoft Office VBA Reference

  <https://docs.microsoft.com/en-us/office/vba/api/overview/> (2022-05-03)

- A collection of Microsoft of administrative scripts for Windows
  - A self unarchiving Windows help file

    <http://download.microsoft.com/download/.NetEnterpriseServer/Utility/1.0/NT5XP/EN-US/netscrpt.exe> (2022-04-22)
  - Collection of Visual Basic scripts for administrative tasks
  - Simply transcribable to ooRexx

    ➔ *Hint:* simply replace the dot (.) in Visual Basic programs with the ooRexx message operator (the tilde: ~)