Consultingwerk
software architecture and development

Platform-independent Word and Excel documents with OpenXML

Peter Judge
peter.judge@consultingwerk.com

Peter Judge

Writing 4GL since 1996, working on a variety of frameworks and applications.

Active member of the OpenEdge community and speaker at international conferences

Focusing on integration with OpenEdge applications, PASOE, software architecture and web technologies

Modernization in Focus

Modernization of Legacy OpenEdge Applications

Deep Technical Expertise

Global IT Partner with Local Presence

More than Consulting – We Deliver Tools & Solutions

Agenda

- **The classic approach**
- OpenXML classes
- .NET Core

Why use Office (Word and Excel)?

- "Cheap" reporting solution, fairly pervasive (100's of millions of users)
  - Good-enough solution for creating documents from application data

- Data exported from browsers, grids or directly from DB tables
- Creation of invoices, quotes, other documents
  - Can contain images, barcodes
- Perform mail merges

The Classic approach in OpenEdge

- Windows-only
- Requires Office installed on the PC or server
- Uses OCX's
  - Has worked since v7
  - Microsoft does provide 64-bit versions
- Documents opened in the Word or Excel application and manipulated there

The Classic approach in OpenEdge

```
define variable chExcel     as com-handle no-undo.
define variable chWorksheet as com-handle no-undo.
define variable chWorkbook  as com-handle no-undo.

create "excel.application" chExcel.

/* Open an Excel document  */
chExcel:Workbooks:Open(FileHelper:FindFile("Test1.xl
sx")).
/* Open Excel maximized */
chExcel:WindowState = -4137.
chExcel:visible = true.

/* Sets the number of sheets that will
be   automatically inserted into new workbooks */
chExcel:SheetsInNewWorkbook = 5.
/* Add a new workbook */
chWorkbook = chExcel:Workbooks:Add().

/* Add a new worksheet as the last sheet */
chWorksheet = chWorkbook:Worksheets(5).
chWorkbook:Worksheets:add(, chWorksheet).
release object chWorksheet.

/* Select a worksheet */
chWorkbook:Worksheets(2):Activate.
chWorksheet = chWorkbook:Worksheets(2).
```

```
/* Rename the worksheet */
chWorkSheet:NAME = "test".

/* Modify the cell's format to Text */
chWorksheet:Cells:NumberFormat = "@".
/* Modify the cell's format to Date */
chWorksheet:Cells:NumberFormat = "m/d/yy;@".
/* Change the cell's color */
chWorksheet:Columns("A:A"):Interior:ColorIndex = 5.

/* Change the cell's format  */
assign
  chWorksheet:Columns("A:A"):Font:ColorIndex = 2
  chWorksheet:Columns("A:A"):Font:Name = "Courier
New".
  chWorksheet:Columns("A:A"):Font:Bold = true.
  chWorksheet:Columns("A:A"):Font:Italic = true.

/* Set underline: StyleSingle = 2 */
chWorksheet:Columns("A:A"):Font:Underline = 2 .
/* Add data */
assign
  chWorksheet:Range("B1"):Value = "Value"
  chWorksheet:Range("B2"):Value = 255
  chWorksheet:Range("B3"):Value = 100
  chWorksheet:Range("B4"):Value = 250
  chWorksheet:Range("B5"):Value = 400
  chWorksheet:Range("B6"):Value = 100
  chWorksheet:Range("B7"):Value = 600.
/* Add a Formula */
chWorksheet:Range("A8"):Value = "Total:".
chWorksheet:Range("B8"):NumberFormat = 0.
chWorksheet:Range("B8"):Formula = "=SUM(B2:B7)".

/* Set horizontal alignment. Right Alignment: -4152
/ Left Alignment :-4131  */
chWorksheet:Range("B:B"):HorizontalAlignment = -
4152.

/* Save the new workbook without displaying alerts
*/
chExcel:DisplayAlerts = false.
chWorkbook:SaveAs(FileHelper:CombinePath(FileHelper:
FindFile("."), "test2.xlsx"),51,,,,,). /* file
format 51 = OpenXML format without macros */

/* Quit Excel */
chExcel:quit().
```

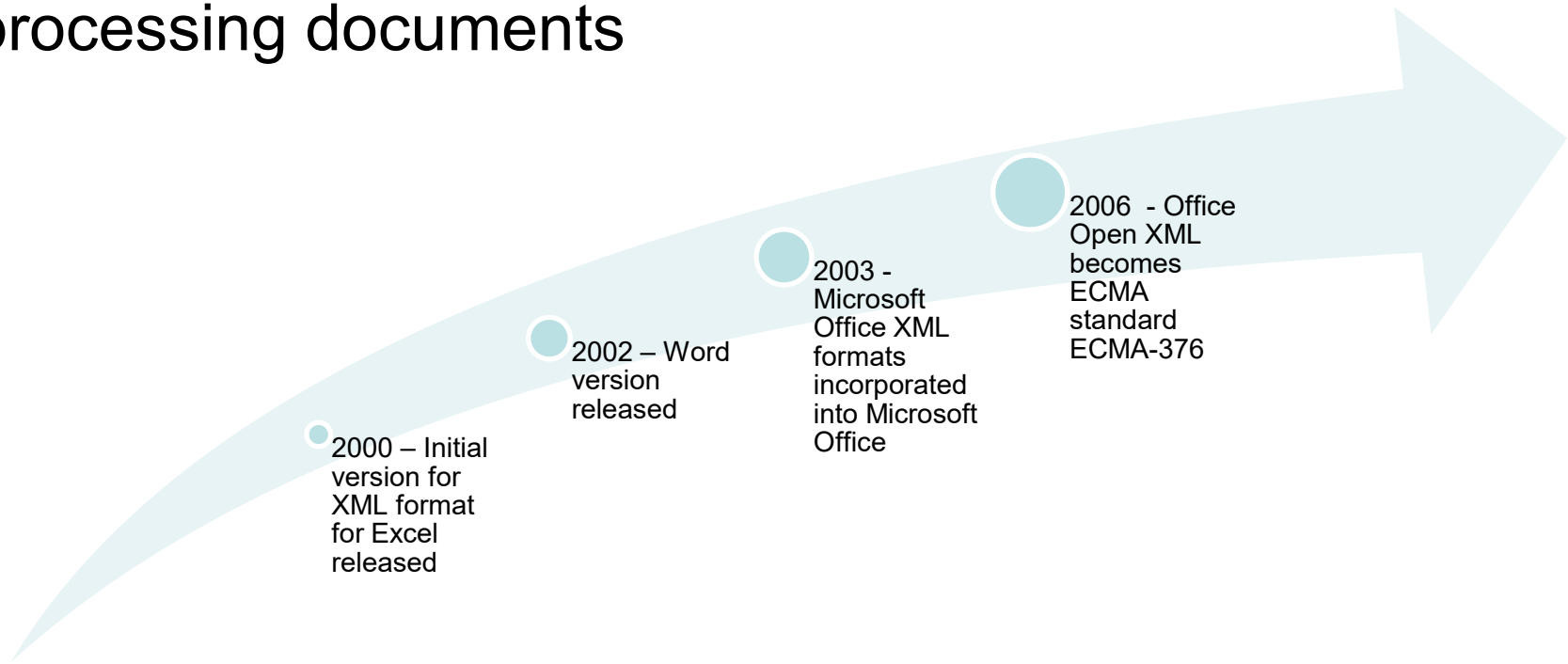
https://community.progress.com/s/article/21671

Agenda

- The classic approach
- **OpenXML classes**
- .NET Core

Office Open XML (OOXML)

- Office Open XML is a zipped, XML-based file format developed by Microsoft for representing spreadsheets, charts, presentations and word processing documents



What is OpenXML

The standard itself is structured into four parts, each of which contains normative as well as informative material

1. Fundamentals and Markup Language Reference (5558 pages) Part 1 of the ISO29500 standard contains definitions for strict conformance as well as the reference material for **WordprocessingML**, **SpreadsheetML**, PresentationML, DrawingML, Shared MLs and Custom XML Schema. It defines every element and attribute including the element hierarchy (parent/child relationships).
2. Open Packaging Conventions standard defines the Open Packaging Conventions (package model, physical package) along with the core properties, thumbnails and digital signatures.
3. Markup Compatibility and Extensibility standard clearly specifies how elements and attributes should be introduced by future versions or extensions of Open XML documents. It describes extension facilities of Open XML documents while also providing a method by which consumers can obtain a baseline version of the OpenXML document (a version without extensions) for interoperability.
4. Transitional Migration Features standard contains definitions for transitional conformance. It defines features for backward-compatibility which are useful for the high-quality migration of existing binary Microsoft Office documents.

Open XML Format SDK

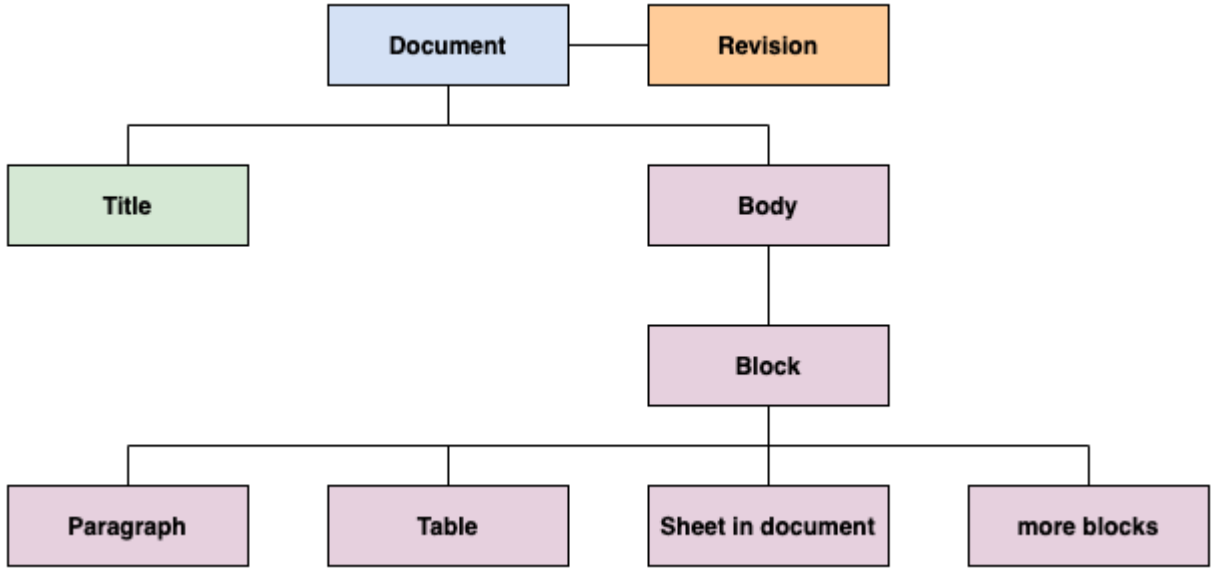
- Microsoft Open XML Format SDK contains a set of managed code (C#) libraries to create, read and update Office Open XML files programmatically
  - Version 1.0 released on June 10, 2008
  - https://github.com/dotnet/Open-XML-SDK
  - https://learn.microsoft.com/en-us/office/open-xml/open-xml-sdk
- No need to have Office installed, no need to use Office 365 web services
- Accessible in OpenEdge via the GUI for .NET bridge

ABL Generation of Office Documents

- The API is quite different from the OCX version
- Some API's not supported by ABL directly
  - Samples based on usage of LINQ
  - Usage of generic methods (on non-generic types)
  - Some samples use latest C# features not available in .NET 4.8 (nullable/non-nullable types)
- Requires development of small .NET Helper Assembly to expose API's compatible to OpenEdge
- .NET performance can be significantly better
- "Save As Pdf" not available

Word documents: top-level structure

- document
- comments
- settings
- endnotes
- ftr (Footer)
- footnotes
- glossaryDocument
- hdr (Header)
- styles



Word documents: document structure

- <r> = "run", which is a region of text with a common set of properties, such as formatting
  - Contains one or more <t> (text range) elements
  - Run properties change formatting
- You will need to read the documentation for the various elements

```
<w:document
xmlns:w="http://schemas.openxmlformats.org/wor
dprocessingml/2006/main">
  <w:body>
    <w:p>
      <w:r>
        <w:rPr>
          <w:b />
          <w:i />
          <w:color w:val="8EFF7C" />
        </w:rPr>
        <w:t>Example text.</w:t>
      </w:r>
    </w:p>
  </w:body>
</w:document>
```

https://learn.microsoft.com/en-us/office/open-xml/word/how-to-create-a-word-processing-document-by-providing-a-file-name?tabs=cs-0%2Ccs-1%2Ccs-2%2Ccs

Document Creation Demo

- Word
  - Create Word 'naked'
    - barcode.p
    - paragraph.p
  - Create Word from bookmarks
    - Receipt.p
    - Invoice / table

- Excel
  - Export to Excel from CSV
  - Create Excel from data

Agenda

- The classic approach
- OpenXML classes
- **.NET Core**

.NET Core

- Announced 2014 as "Cloud-optimized .NET Framework"
- First RC was named .NET Core 5.0 to indicate continuity with .NET Framework 4.x releases
- However, it was released as .NET Core 1.0
- **Accepts breaking changes compared to .NET Framework**
- With release 5.0 “Core” was dropped from the name: .NET 5.0
- Performance improvements in .NET 6.0, further performance improvements in .NET 7.0
- Related to the Mono framework (.NET port to Linux)

.NET / .NET Core / .NET Framwork

- Don’t shoot the messenger!

.NET Framework is not .NET

.NET is not .NET Framework

.NET Core

- Large parts of the community refer to .NET 6.0 as .NET Core 6.0 – even when Microsoft is no longer using that name
- It‘s just human to distinguish .NET Core vs. .NET Framework
- Even Progress Software has named the startup parameter to enable support for .NET 6.0 in the AVM

-clrnetcore

.NET Core Details

- Platform independent (Windows/Linux/macOS x86/x64, ARM)
- Open Source (MIT/Apache2) – https://github.com/dotnet/core
- C# / F# / VB.NET (with limitations)
- Visual Studio / Visual Studio Code / Command Line (dotnet new|build|run)
  Focus on CLI tools, micro services and Web development

OpenEdge and .NET

- 2003 / OpenEdge 10.0: Introduction of OpenClient for .NET – wow that was quick!
- 2007 or 2008 / OpenEdge 10.2A: Introduction of OpenEdge GUI for .NET – ability to use .NET WinForms UI from OpenEdge GUI client, CLR 2.0
- 2009 / OpenEdge 10.2B: GUI bridge opened to non-GUI use-cases
- 2011 / OpenEdge 11: CLR 4.0 for GUI for .NET (CLR 2.0 cannot be used anymore)
- 2023 / OpenEdge 12.7: **Optional** use of .NET 6 runtime, .NET 4.8 default runtime, OpenEdge 12.8: Use of .NET 8

Using .NET Core in OpenEdge

- Starting OpenEdge 12.7 OpenEdge supports .NET 6.0 alternatively to the .NET Framework
- Supported in OpenEdge GUI client (prowin, prowin32), Web Client, Character Client and PASOE
- By default, .NET Framework 4.8 is used
- Use **-clrnetcore** startup parameter to use .NET 6.0 **instead**!
- No parallel use of .NET Framework and .NET Core

.NET Core for OpenEdge – why?

- Latest version of .NET
- Future-proofing application development
- It is expected that in the foreseeable future modern libraries will be increasingly available for .NET Core – and no longer the “legacy” .NET Framework
- Faster …
- And …

.NET Core support on Linux

- OpenEdge 12.8 extended support for .NET Core to Linux
- Allows usage of API's in OpenEdge Linux runtimes
  - TTY Client
  - Batch Client
  - PASOE

- Certain classes of the .NET Framework (eg image processing) are *not* in .NET Core

.NET Versions

- Support for .NET 8.0 added in the 12.8 series

```
//$DLC/bin/Progress.clrbridge.netcore.runtimeconfig.json

{
  "runtimeOptions": {
    "tfm": "net8.0",
    "frameworks": [
      {
        "name": "Microsoft.NETCore.App",
        "version": "8.0.0"
      },
      {
        "name": "Microsoft.WindowsDesktop.App",
        "version": "8.0.0"
      }
    ],
    "configProperties": {
      "System.Reflection.Metadata.MetadataUpdater.IsSupported": false
    }
  }
}
```

- Kbase: Which .NET Versions are supported with OpenEdge? https://community.progress.com/s/article/000054406

More info about .NET Core

- Consultingwerk YouTube Webinar
  Using .NET Core and OpenEdge
  https://youtu.be/AJGC_07-Yv8?si=XKLaINR16oLp-r1B

Linux Demo

- Same programs as earlier, remove the "open in Word" code.

In closing

- With the ability to use .NET Core on Linux in 12.8, we can generate Office documents from ***all*** supported OpenEdge platforms
  - Use then AppServer to offload report and document generation
  - Move away from an old technology
  - Does need some custom .NET code to make the integration easier from ABL
- Performance for larger docs vastly better
- Third-party libraries or tools may be required, eg for PDF conversion

Questions

Subscribe Now to Consultingwerk's Official Newsletter!

www.consultingwerk.com/newsletter

or marketing@consultingwerk.com


Consultingwerk
software architecture and development