OOXML Transitional Migration Features

A Technical Reference

Companion document to "A Standard in Name Only: What OOXML Transitional Tells Us About Format Sovereignty".

All element names and section references in this document have been verified against ISO/IEC 29500-1:2016 and ISO/IEC 29500-4:2016.

Introduction

ISO/IEC 29500 — the international standard underlying Microsoft's `docx`, `xlsx`, and `pptx` file formats — is published in four parts, totalling several thousand pages. **Part 1, *Fundamentals and Markup Language Reference***, contains the Strict conformance definition: the clean, modern document format that an independent implementer could reasonably be expected to support. **Part 4, *Transitional Migration Features***, contains the rest: a catalogue of compatibility elements, deprecated constructs, platform-specific behaviours, and references to undocumented quirks of Microsoft Office versions from the 1990s.

The Transitional class exists to ensure that documents converted from the old binary `doc`, `xls` and `ppt` formats — billions of files accumulated over two decades of Microsoft Office's market dominance — can be represented in XML without loss. In practice, Transitional is also what Microsoft Office itself produces: Strict has never been the default in any version of Microsoft Office, and the option to save in Strict format is inconsistently available across editions of Microsoft 365 — present in the installed desktop applications, absent from the browser-based versions.

This document catalogues the principal categories of Transitional features and explains the structural problems they introduce for any party attempting to implement, validate, or rely on OOXML as an open standard.

How the Standard Describes Its Own Compatibility Features

Before cataloguing the Transitional features, it is worth noting how Part 4 describes them in its own words. The introductory text to the compatibility settings section (§14.8.3) states that "a number of the settings reference specific applications and specific versions of those applications" and that "this is solely for backward compatibility reasons." The spec is candid: large parts of Part 4 are not so much specifications as references to the behaviour of identified commercial products.

More remarkably, the standard repeatedly characterises the behaviour it is mandating as

incorrect. Section 14.8.3.5 (`cachedColBalance`) opens with the phrase "applications shall incorrectly calculate the height of a paragraph" and adds the editorial guidance: *"It is recommended that applications not intentionally replicate this behavior; it is maintained only for compatibility with existing documents from a legacy application."* Section 14.8.3.55 (`useWord97LineBreakRules`) prefaces its specification with the admission that the calculations it mandates "would not normally be considered correct." Section 14.8.3.54 is titled `useWord2002TableStyleRules` with the descriptive subtitle "Incorrectly Display Top Border of Conditional Columns." Section 14.8.3.4 (`autoSpaceLikeWord95`) is titled "Incorrectly Adjust Text Spacing for Specific Unicode Ranges."

A standard that openly describes its own mandated behaviour as incorrect, and that advises implementers against reproducing parts of itself, is in a curious position. These are the actual words of ISO/IEC 29500-4.

Categories of Transitional Features

Legacy application emulation

A substantial body of Transitional elements instructs modern applications to emulate the behaviour of specific old versions of Microsoft Word, MacWord, or WordPerfect. These elements appear inside the `<w:compat>` block in `settings.xml`. Verified examples from §14.8.3 include:

- `<w:autoSpaceLikeWord95>` (§14.8.3.4) — space punctuation as Word 95 did; the section title characterises this behaviour as "incorrectly" adjusting text spacing
- `<w:footnoteLayoutLikeWW8>` (§14.8.3.20) — the element name references Word 97/2000 ("WW8") footnote behaviour, although the descriptive title reveals that the actual mandated behaviour concerns section breaks in the presence of footnotes
- `<w:lineWrapLikeWord6>` (§14.8.3.25) — wrap lines using the Word 6.0 algorithm
- `<w:mwSmallCaps>` (§14.8.3.26) — use a specific small caps algorithm (the section defines an opaque numerical sequence whose only justification is matching a particular legacy application's output)
- `<w:shapeLayoutLikeWW8>` (§14.8.3.35) — position shapes using Word 2000 logic
- `<w:splitPgBreakAndParaMark>` (§14.8.3.38) — handle page breaks in a particular legacy manner

- `<w:suppressTopSpacingWP>` (§14.8.3.43) — apply a static baseline-to-baseline distance calculation inherited from WordPerfect
- `<w:truncateFontHeightsLikeWP6>` (§14.8.3.46) — use truncated integer division for font height conversion, mimicking WordPerfect 6.0
- `<w:useWord97LineBreakRules>` (§14.8.3.55) — apply inter-character spacing rules whose own specification text concedes "would not normally be considered correct"
- `<w:useWord2002TableStyleRules>` (§14.8.3.54) — incorrectly display the top border of conditional columns
- `<w:wpJustification>` (§14.8.3.56) — justify text using a specific WordPerfect-derived width calculation involving the magic number 281/7200

The specification of these elements is largely by reference: "emulate the behaviour of application X, version Y." The behaviour of those applications was never published. An independent implementer must therefore reverse-engineer software from the 1990s — including its bugs and edge cases — in order to render conformant documents correctly.

Vector Markup Language (VML)

VML was submitted by Microsoft to the W3C in May 1998 as a proposed standard for vector graphics on the web. The W3C considered it alongside Adobe and Sun's competing PGML submission, and ultimately chose neither: SVG was developed instead and reached Recommendation status in 2001. VML was therefore not a "pre-standard" format in any neutral sense — it was a candidate that the relevant standards body rejected.

Despite this, VML persists inside OOXML Transitional because documents converted from `.doc` files contain VML markup, and Microsoft Office continues to emit it in certain contexts. Implementations must support both VML and its modern replacement, DrawingML, to handle real-world files. Representative VML elements include `<v:shape>`, `<v:rect>`, `<v:oval>`, `<v:line>`, `<v:textbox>`, and `<v:imagedata>`. Section 14.8.4.1 of Part 4 (`relyOnVML`) further specifies how VML output is to be preserved when WordprocessingML content is saved as a web page.

Platform-specific elements

A number of Transitional features assume specific Microsoft platform technologies. The most explicit example is `<w:usePrinterMetrics>` (§14.8.3.52), which directs applications to use printer-specific metrics to lay out documents. The non-normative note accompanying this section illustrates the implementation by referring directly to

the Windows API: *"On the Windows platform, you can use the GetDeviceCaps function to retrieve device-specific information for the specified printer."* The standard does not claim that GetDeviceCaps is mandatory — but it offers no implementation guidance for any other platform, and the algorithm it describes assumes a Windows GDI device context.

Other platform-bound features include ActiveX control embedding (Windows-only by design), OLE object embedding via the Component Object Model, and XSLT processing that historically assumed the Microsoft XML parser (MSXML).

Behaviours mandated as incorrect

A distinct category of Transitional elements does not emulate a specific legacy application but instead mandates calculations or rendering behaviours that the standard itself describes as incorrect. The clearest examples:

- `<w:cachedColBalance>` (§14.8.3.5) — mandates that applications "shall incorrectly calculate the height of a paragraph" for column balancing
- `<w:useWord97LineBreakRules>` (§14.8.3.55) — mandates inter-character spacing calculations that the spec acknowledges "would not normally be considered correct"
- `<w:useWord2002TableStyleRules>` (§14.8.3.54) — titled "Incorrectly Display Top Border of Conditional Columns"
- `<w:convMailMergeEsc>` (§14.8.3.6) — mandates that backslash-quotation delimiters be silently converted to double quotation marks during mail merge, a transformation that changes the data being merged

These are not legacy compatibility features in any neutral sense. They are calculations the standard requires implementations to perform, while simultaneously noting that the calculations are wrong.

Deprecated document properties and platform features

Beyond the compatibility settings, OOXML Transitional retains properties carried forward from legacy formats — for example, document settings related to SmartTags (a deprecated Office feature), Windows-specific font embedding, XSLT processing assumptions, and the 1900-leap-year date system discussed below.

The 1900 Leap Year and the Architecture of a Mandated Bug

The 1900 leap year bug is the most widely cited defect in OOXML Transitional, but its

treatment in the standard is more interesting than commonly reported. Part 1, §18.17.4.1, defines two date systems for SpreadsheetML:

"In the 1900 date system, the lower limit is January 1st, 0001 00:00:00, which has a serial date-time of -693593. The upper-limit is December 31st, 9999, 23:59:59.999, which has a serial date-time of 2,958,465.9999884. The base date for this system is 00:00:00 on December 30th, 1899, which has a serial date-time of 0."

The base date of the 1900 date system is **December 30, 1899** — not, as the system's name might suggest, January 1, 1900. The reason is the leap year bug: to preserve compatibility with Lotus 1-2-3, which had assumed that 1900 was a leap year, the serial date arithmetic must include a phantom February 29, 1900. Shifting the base date back by two days accommodates this phantom day while keeping the serial date-time of, for example, January 1, 1900 at the expected integer value.

What is striking is that the standard nowhere states explicitly that 1900 is to be treated as a leap year. The bug is encoded structurally — through the choice of base date and the worked examples that follow it. A reader of §18.17.4.1 who is not already familiar with the history will find a coherent-looking set of date system definitions with no flagged anomaly. The bug is present in the arithmetic without being named.

The 1904 date system, defined alongside the 1900 system in the same section, has a base date of January 1, 1904 — the date its name implies — because no Lotus 1-2-3 compatibility constraint applied to that system. The asymmetry is its own evidence.

The practical consequence is that every `.xlsx` file produced with default settings encodes Lotus 1-2-3's 1980s arithmetic error. Date calculations spanning February 1900 produce incorrect results. The standard requires this behaviour and never acknowledges it as a defect.

Structural Problems

Namespace pollution

Transitional features share XML namespaces with Strict-conformant features. There is no clean syntactic separation. Validating whether a given document is genuinely Strict-conformant — as opposed to Transitional with no Transitional elements happening to be present — requires walking the entire document tree and checking each element against the Strict subset.

Specification by reference to undocumented behaviour

Many Transitional elements are specified in terms such as "lay out like Word 95" or "calculate spacing as WordPerfect did." These are not formal specifications. They are references to the observable behaviour of commercial products whose source code,

layout engines, and algorithms were never published. Any implementer attempting to support these elements must work from reverse-engineering, leaked documentation, or trial and error against the original applications.

The result is that the parts of the standard most likely to appear in real-world documents are also the parts least amenable to independent implementation.

Deprecation without removal

Elements marked as deprecated or legacy remain in the specification indefinitely. There is no scheduled deprecation timeline, no version of the standard in which the Transitional class is to be withdrawn, and no mechanism by which the standard might eventually converge on the Strict subset. Because Microsoft Office continues to produce Transitional output, removing these elements from the standard would invalidate documents that the dominant producer of OOXML files continues to generate.

Self-acknowledged incorrectness

A subset of Transitional elements are mandated even though the standard itself describes their behaviour as incorrect. In the case of `<w:cachedColBalance>` (§14.8.3.5), the spec explicitly recommends that implementations *not* reproduce the behaviour: *"It is recommended that applications not intentionally replicate this behavior; it is maintained only for compatibility with existing documents from a legacy application."* An international standard that advises against following parts of itself is in an unusual position.

The Conformance Class Problem

The standard defines two conformance classes with very different practical statuses.

Strict conformance excludes the Transitional features catalogued above. It is the intended "clean" version of OOXML. In practice, Strict has never been the default save format in any version of Microsoft Office, read/write support arrived only with Office 2013, and the option to save in Strict format is inconsistently available across editions of Microsoft 365 — present in the installed desktop applications, absent from the browser-based versions. Cross-edition inconsistency is a long-standing pattern in Microsoft Office's history: the macOS version has historically provided a different set of options from the Windows version. Strict conformance therefore exists primarily as a specification artefact: a definition that no major producer generates by default and that Microsoft itself supports unevenly across the products it sells under the same name.

Transitional conformance includes all the legacy elements. It is what Microsoft Office actually produces, what users actually receive, and what any competing implementation must support to be useful in practice. From the point of view of an implementer, Transitional is the standard.

The practical consequence is that the parts of OOXML that exist on paper as a "clean modern standard" are not the parts that real documents use, and the parts that real documents use are precisely the parts that are most poorly specified, most platform-dependent, and most reliant on knowledge of a single vendor's implementation history.

Illustrative Examples

`<w:footnoteLayoutLikeWW8>` (Word)

```
<w:compat>
  <w:footnoteLayoutLikeWW8/>
</w:compat>
```

The element name references the layout behaviour of Word 97/2000 ("WW8" being the internal version identifier for Word 97). The actual algorithms used by those versions of Word — including edge cases, line-break behaviour, and interactions with page geometry — were never formally documented. An implementer must reverse-engineer Word 97 to comply.

The 1900 date system (Excel)

```
<workbookPr date1904="false"/>
```

When `date1904="false"` (the default), implementations must treat dates according to the 1900 date system defined in Part 1, §18.17.4.1. That system has a base date of December 30, 1899 — a structural encoding of the phantom February 29, 1900 inherited from Lotus 1-2-3. Spreadsheets performing date arithmetic across February 1900 will produce incorrect results, and the standard requires this incorrect behaviour without explicitly acknowledging it as a defect.

VML in converted documents

```
<w:pict>
  <v:shapetype id="_x0000_t75" coordsize="21600,21600"
               o:spt="75" o:preferrelative="t"
               path="m@4@5l@4@11@9@11@9@5xe" filled="f" stroked="f">
    <v:stroke joinstyle="miter"/>
    <v:path gradientshapeok="t" o:connecttype="rect"/>
  </v:shapetype>
</w:pict>
```

VML markup of this kind appears in documents converted from `.doc` format. Implementations supporting only the modern DrawingML graphics system will fail to render such content.

`<w:cachedColBalance>` — a mandated miscalculation

```
<w:compat>
```

```
  <w:cachedColBalance/>
</w:compat>
```

The opening sentence of §14.8.3.5 states that this element "specifies whether applications shall incorrectly calculate the height of a paragraph for the purposes of column balancing." The accompanying guidance recommends that applications not intentionally replicate the behaviour. An implementer is thus instructed to implement an incorrect calculation, knowing it to be incorrect, in order to remain compatible with documents produced by a legacy application.

Implications for Interoperability

The Transitional features create four distinct implementation problems.

Implementation burden. Any third party wishing to read or write OOXML files in their real-world form must implement not only the modern Strict subset but the much larger Transitional surface area, including elements specified by reference to undocumented legacy behaviour and elements whose required behaviour is described as incorrect.

Testing complexity. Validating correctness requires comparing output against Microsoft Office's behaviour — including the behaviour of long-discontinued versions of Office that are referenced in the standard. No formal test suite can capture the totality of "what Word 97 did."

Standard fragmentation. Documents produced by Microsoft Office are not Strict-conformant. The value of Strict conformance to procurement officers, archivists, or policy-makers specifying "ISO/IEC 29500" is therefore limited: the conformance class they may have in mind is not the one that real documents use.

Indefinite legacy. The Transitional features have no deprecation timeline. They remain in the standard as long as Microsoft Office continues to emit them, which is to say indefinitely.

Conclusion

The Transitional Migration Features encode the central design choice of OOXML as an international standard: rather than specifying a clean, implementation-neutral document format, the standard formalises the existing behaviour of one company's product line, including thirty years of accumulated compatibility decisions, platform assumptions, and inherited bugs — some of which the standard itself acknowledges as incorrect and advises implementers not to reproduce.

This is the basis for the long-standing criticism that OOXML is a "standard" in name only. The ISO label is real. The two-part conformance structure is real. But the standard

most users actually encounter — and the one that Microsoft Office produces by default and supports most consistently across its product line — is the one that perpetuates legacy behaviour, depends on undocumented properties of 1990s applications, structurally encodes arithmetic errors inherited from the 1980s, and cannot be cleanly implemented by any party without privileged access to Microsoft's institutional knowledge.

For public administrations, archivists, and procurement officers weighing format policy, the implication is direct: specifying OOXML in a tender does not deliver the vendor-neutral interoperability that an open standard is meant to provide. The OpenDocument Format (ODF, ISO/IEC 26300) was designed from the outset to avoid these problems and remains the only widely deployed document standard that meets the substantive — not merely formal — definition of openness.