Editor: Barry Leiba • leiba@watson.ibm.com

OpenDocument Format

The Standard for Office Documents

Rob Weir • *IBM*

OpenDocument Format (ODF) is an XML-based open standard file format for office documents, such as spreadsheets, text documents, and presentations. ODF is application-, platform-, and vendor-neutral and thereby facilitates broad interoperability of office documents.

Although personal productivity applications (PPAs) and their associated file formats have been around for many years, they haven't been the subject of standardization until very recently. We can better understand the current importance and relevance of standardization in this area if we first consider the technological, historical, and economic context of electronic documents.

After more than a decade of evolution, PPA functionality has converged on a set of capabilities that are conventionally expressed via three application types: a word processor, a spreadsheet, and a presentation graphics application. Although Microsoft, Corel, Sun, IBM, and Google all offer PPAs, as do open source vendors, the difference in functionality among these offerings is relatively minor compared to the difference in functionality among the three application types. In other words, the difference between Microsoft Word and Corel WordPerfect is minor compared to the difference between Microsoft Word and Microsoft Excel. Given this convergence in functionality within each application type, we can now describe their core commonalities in an open, standardized document format.

Historically, we weren't so fortunate. When we examine the history of PPAs, we see that the general practice was for each vendor to define a file format specifically and exclusively for use with its own particular product. So, WordPerfect had its own format, as did WordPro, Word, WordStar, XyWrite, Writing Assistant, StarOffice, and so on. From a technical perspective, in the days when RAM and processing cycles were precious, this was often the pragmatic choice. An ad hoc file format, designed specifically for a given application, will typically parse faster and consume less memory than a general-purpose one. In fact, to optimize performance, some early formats were little more than a direct serialization of their application's internal data structures.

Public availability of documentation for these early file formats varied. Some vendors made their formats available on demand or published them in book form. Others never released their formats. Still others released them under restrictive licenses that limited use by competitors. Interoperability among word processors, where it existed at all, was hard-won and often accomplished by reverse engineering and trial and error.

And so it was, through much of the 1980s and 1990s. Although the formats weren't interoperable or specified in open standards, this was tolerable because office documents were primarily exchanged as printed copies. Where exchanges occurred electronically, they were most often between known parties, such as workers in the same office, using the same software. Interoperability in such cases was trivially obtained because document authors could make strong, valid assumptions about the receiving party's software and platform. In other words, if everyone you correspond with uses the same version of the same word processor running on the same version of the same operating system configured the same way with the same fonts installed, then, in theory, electronic document exchanges should be straightforward. But reality is rarely so cooperative.

1089-7801/09/$25.00 © 2009 IEEE

From the Department Editor

For many years, when you used a word-processing application, your files were stored in a format that was fully understood only by the application you used. WordPerfect stored WordPerfect files, Microsoft Word stored Word files, and so on — and, while developers often included ad hoc support for their competitors' formats, that was a hit-and-miss thing, vulnerable to changes in proprietary formats. The same went for presentation slides and spreadsheets. We all have experience with the results of this, with the difficulties in exchanging files between different applications.

ODF — OpenDocument Format — addresses this problem by providing a standard format for storage and exchange of office documents. It works on Windows, MacOS, Linux, and other platforms, and it works for any application whose vendor chooses to implement it. In this issue's "Standards" department, IBM's Rob Weir, who worked on the ODF standard, will take us through some of its history and details.

— *Barry Leiba*

Why ODF Now?

In recent years, our complacency with a lack of office-document standards has been shattered. Document formats, which historically were unseen and unspoken of, are now hotly debated. Governments throughout the world now mandate or recommend the use of ODF.[1] Corporations are investing significant time and effort to further ODF adoption or, in some cases, prevent it. Why the increased emphasis on document format standards?

Several factors are at play here.

First, when the World Wide Web became a mainstream success, new document distribution patterns emerged. Authors could now easily post documents online for anyone to download, but they could no longer assume that the receiving party was using the same software stack. The receiver's capabilities were unknown and, in principle, unknowable. In such an environment, we can achieve interoperability only through deliberate engineering efforts, not by merely assuming a universe of homogeneous systems. The primary way to do this is by creating and adopting standards.

Second is the economic impact of "vendor lock-in" and the deleterious effects that ensue. If a document format is designed traditionally — that is, to closely reflect the internal data structures of a single vendor's applications — then documents created using those applications will only work, or work well, with those applications. As the user continues to create and accumulate documents in that proprietary format, his or her ability to evaluate and migrate to alternative PPAs in the market diminishes due to substantial switching costs. The presence of high switching costs is the essence of vendor lock-in and is a significant challenge from the perspective of the user, who now faces effectively diminished options; the potential competitor, who finds it more difficult to enter a market dominated by an incumbent with an established lock-in; and public policy, which has a long-standing practice of encouraging competition. One fundamental way to reduce the strength of vendor lock-in and encourage substitutability of equivalent goods is by adopting and using open standards that encourage interoperability.

The debate these issues generate has been amplified by its predominate impact on public administrations, in which the document-based bureaucracy, along with being perennially budget-challenged, is constrained by public accountability, the need to provide equal access to citizens, and the often legally mandated duty to preserve documents according to defined archival policies. Increasingly, governments communicate with citizens electronically, both in publishing government documents and in receiving documents from the public. When a government adopts, by deliberate policy or inertial passivity, a vendor's proprietary document format — which then promotes the exclusive use of that same vendor's products — this requirement then trickles down to all those who wish to deal with that government. This unintended consequence thus becomes an impediment to those in society least able to afford the proprietary software. For example, in the aftermath of Hurricane Katrina, the US Federal Emergency Management Agency (FEMA) set up a Web site for disaster victims to apply for federal assistance. However, the Web site was designed to work only with Microsoft Internet Explorer running on Microsoft Windows, rather than strictly following the W3C's HTML standard, which would have worked everywhere. So, hurricane victims attempting to connect from a friend's Macintosh or the library's Linux machine were unable to apply for aid online.[2] Dependency on a single vendor's proprietary formats is bad economics as well as bad policy.

Document Format Technical Requirements

A standard office format must support the gamut of current uses. Office documents, as used today, range from unstructured, free-form documents, to semi-structured memos, to strongly structured forms. PPA users range in awareness from novices with little or no conception of formal document structure and the benefits of style and content separation to highly sophisticated users creating highly structured documents as part of a larger document-processing pipeline. A document format must facilitate and encourage modern best practices in document design, but it

should still enable ad hoc use by novice users, who conceptualize documents merely in WYSIWYG terms.

A document format must deal with the complexity of modern PPAs, including a lengthy list of conventional features such as headers and footers, footnotes, tables of contents, revision tracking, and so on. It should define an explicit encoding for such common features and allow extensibility for more arcane features that only a single implementation might use.

Electronic documents exist in an international context and need to adapt to international linguistic, cultural, and business customs, both current and historical. This encompasses issues such as differing character sets, bidirectional text, ruby (furigana) annotations, varying ways of representing currency symbols, dates, and numbers, varying rules for sorting (collating) text, and so on.[3] Because these textual conventions have evolved over centuries of uncoordinated scribal practice, they don't easily reduce to a single, clean, logical formulation.

A document format must consider application and data security as well as issues such as document encryption, digital signatures, and the ramifications of executable code embedded within documents via scripts or macros. For example, all executable code in a document must be clearly declared as such to enable third-party antivirus utilities to find and scan the code for threats.

Users of electronic documents vary in their abilities. For instance, users with visual impairments might require "assistive technology," such as screen readers, to read and edit documents.[4] To be compatible with such devices, a document format must ensure that it can encode textual annotations for any content that would otherwise be purely graphical. An embedded image, for example, must allow an associated text description. Similarly, a document format should facilitate easy global and local navigation — for instance, by encouraging the use of structured tables with explicitly declared row and column headers. Defining the optimal characteristics of "accessible" documents is a crucial and ongoing task.

Note also that not all document consumers are human users. Increasingly, we see that documents are read and even created by software automation. From the simple text scanner for a search engine's indexer to more sophisticated information extraction and analysis applications, the trend is for a document to be read many more times, and by many more applications, than were ever involved in creating it. So, a document format must be platform independent as well as adaptable to disparate application and machine types. Documents are not only for desktop PC use. They are read and written on "headless" servers without any user interface; edited on Web-browser-based word processors; or even viewed and edited on mobile phones and other small form-factor devices with constrained user interfaces. A document format must accommodate all the places and ways in which we use documents.

A History of ODF

The ODF standard was created and is maintained by the ODF Technical Committee (TC) within the Organization for the Advancement of Structured Information Standards (OASIS). See the "ODF Resources" sidebar for links to more information.

OASIS formed the ODF TC in late 2002; it includes representatives from commercial and open source software publishers, government, and academia. The TC's charter aimed to create an open, XML-based file format specification for office applications (http://lists.oasis-open.org/archives/tc-announce/200211/msg00001.html). The file format needed to

- be suitable for office documents containing text, spreadsheets, charts, and graphical documents;
- be compatible with XML v1.0 and W3C namespaces in XML v1.0 specifications;
- preserve the structure of the document to allow re-editing (for example, footnotes must be stored as structured footnotes, not just as text in the document that looks like a footnote);
- be friendly to transformations using the W3C's Extensible Stylesheet Language (XSLT) or similar XML-based languages or tools;
- keep the document's content and layout information separate to enable independent processing; and
- "borrow" from similar, existing standards wherever possible and permitted.

Sun Microsystems contributed its specification of the XML format that OpenOffice.org uses because it was close to meeting these requirements already, and the TC used this as a starting point to develop ODF.

OASIS published ODF 1.0 in May 2005; the International Organization for Standardization/International Electrotechnical Commission ratified it in May 2006 as ISO/IEC 26300:2006 (www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=43485).

Promotion has occurred via the OpenDocument Format Alliance[5] (not formally associated with OASIS or the OASIS ODF TC) and the OASIS ODF Adoption TC.

OASIS published a minor release, ODF 1.1, in February 2007 to address a few accessibility issues identified during an early ODF deployment by the Commonwealth of Massachusetts.

The OASIS ODF Interoperability and Conformance TC was created in late 2008 to facilitate interoperability among ODF implementations, with plans to develop an ODF

ODF Resources

For definitive information on OpenDocument Format (ODF), see the Web page for the Organization for the Advancement of Structured Information Standards (OASIS) OpenDocument Format Technical Committee (ODF TC): www.oasis-open.org/committees/tc_home.php?wg_abbrev=office.

You can download the current specification (ODF 1.1) here: http://docs.oasis-open.org/office/v1.1/OS/OpenDocument-v1.1.pdf.

The TC is co-chaired by Michael Brauer (michael.brauer@sun.com) and Rob Weir (robert_weir@us.ibm.com).

The TC invites public comments on ODF via its public comment list: www.oasis-open.org/committees/comments/index.php?wg_abbrev=office.

The ODF Adoption TC maintains the OpenDocument.xml.org Web site: http://opendocument.xml.org.

Whitepapers and case studies related to ODF adoption are available at the ODF Alliance's Web site: www.odfalliance.org.

J. David Eisenberg has written a book, *OASIS OpenDocument Essentials*, which he makes freely available online: http://books.evc-cit.info/.

The ODF Toolkit Union has open source libraries for manipulating ODF content here: http://odftoolkit.org.

conformance test suite and host interoperability "plugfests," among other activities.

Application and Tool Support

Most leading word processors support ODF, either natively "out of the box" or via freely downloadable extensions or plug-ins. Commercial applications supporting ODF include traditional desktop word processors such as Microsoft Office, Corel WordPerfect, and Lotus Symphony, as well as Web-based word processors such as Google Docs & Spreadsheets and Zoho Writer. Open source implementations of ODF include OpenOffice.org, KOffice, AbiWord, and Gnumeric.

A wide variety of programming libraries are also available for manipulating ODF content via Java, Python, or C#. Additionally, Sun and IBM recently created a new open source community, the ODF Toolkit Union (http://odftoolkit.org), to give developers tools to work with ODF documents.

ODF Adoption

ODF has come along when awareness, especially within public administrations, about the importance of open standards and avoiding vendor lock-in is high. In fact, ODF's existence and promotion has helped raise general awareness about these topics.

Several policy decisions at various government levels have recommended or even mandated using ODF, including 16 national governments and eight provincial governments (South Africa, Belgium, Germany, Brazil, Croatia, Denmark, France, Malaysia, the Netherlands, Norway, and Uruguay, to name a few).

Salient Technical Features

An ODF document is typically stored and distributed in a container file consisting of a .zip archive that includes an XML manifest file, one or more content XMLs, and associated binary content, such as images or other embedded media. Using .zip was the logical choice because it was already widely supported, both in stand-alone tools and runtime libraries, reducing the effort required to add ODF support to an application. The .zip compression is also beneficial. Although some might criticize markup languages such as XML for increased verbosity in the data representation, once ODF's XML is compressed into a .zip archive, most documents are smaller than they would be in an equivalent proprietary binary format such as Microsoft Word's .doc format.

Each ODF file contains four key embedded XML files:

- manifest.xml, which is the table of contents for the ODF container and lists all contained files keyed by MIME content type (for example, "text/xml" or "image/png");
- meta.xml, which contains document-level metadata, including the bibliographic fields defined by the Dublin Core element set, such as creator, title, subject, and language;
- styles.xml, which contains the style definitions for the document, such as what text attributes should be used for "header 1" style versus what attributes should be used for "normal" text; and
- content.xml, which contains the document's structured content, including paragraphs, lists, tables, frames, and images.

ODF's attribute vocabulary is based on that used in the W3C's XLS Formatting Objects (XSL:FO) standard (www.w3.org/TR/xsl11). So, a text style definition for text that's in 20-point, red, Courier font would be defined as

```
<style:text-properties
fo:color="#ff0000"
style:font-name="Courier New"
fo:font-size="20pt"/>.
```

Mathematical equations in ODF are defined using the W3C's MathML vocabulary (www.w3.org/TR/REC-MathML), and forms are defined using the W3C's XForms standard (www.w3.org/TR/xforms).

Structurally, text paragraphs are delimited using <p> elements, similar to HTML. Hypertext links use the W3C's XLink standard (www.w3.org/TR/xlink). So, a default style

paragraph with a hyperlink to a Web page would look like this:

```
<text:p text:style-
name="default">More
information on ODF can be
found
<text:a xlink:type="simple"
xlink:href = "http://open
document.xml.org/">here</
text:a>.</text:p>
```

More complicated constructs are possible, such as multilevel lists, merged cells in tables, and vector graphics. But the principles remain the same: ODF is application- and vendor-neutral and uses existing standards where practical.

ODF Futures

The OASIS ODF TC is currently completing work on its draft of ODF 1.2, which we hope will be ready for formal public review and approval as an OASIS standard in mid-2009. ODF 1.2 will focus mainly on

- the addition of an RDF/XML and OWL-based metadata framework to allow metadata annotations of ODF content at a fine-grained level, which will facilitate applications such as semantic tagging, real-time collaborative editing, and document compositing from shared fragments;
- the specification of a detailed expression language for spreadsheet formulas, called OpenFormula, which contains hundreds of commonly used logical, mathematical, financial, and scientific functions; and
- additional enhancements to further increase accessibility.

In a parallel effort, as ODF 1.2 is completed, the ODF TC has created an "ODF-Next" requirements subcommittee to collect, classify, and prioritize feature proposals for subsequent ODF versions. You can submit suggestions to the ODF comment mailing list at www.oasis-open.org/committees/comments/index.php?wg_abbrev=office.

Although the initial versions of the ODF standard have focused on encoding the storage format for the three conventional PPA application types, ODF isn't limited to these uses. Its fundamental building blocks — a packaging format for bundling multiple XML files and associated media, text structure and formatting, vector graphics, and mathematical equations — are also applicable to a wider range of application types, such as project management, outlining, mind-mapping software, or wikis.

The conventional WYSIWYG word processor could be nearing the end of its useful lifetime. ODF might evolve to take on greater sophistication in the area of semantic encoding, with facilities to let authors capture, in a structured way, more of what they're thinking. Human thought is far too rich and diverse to be captured merely as bold, italic, or underlined. An allowance for semantic layers could let authors encode not just their assertions but also their judgments, estimations of certainty and doubt, facts versus opinions, provenance, authority, and so on in a way that would better lend itself to visualization, mining, and analysis. The challenge, which we eagerly anticipate, is to evolve ODF in a direction that embraces these (and other) possibilities.

References

1. "Governments Increasingly Turn to OpenDocument Format as ODF Alliance Marks Unprecedented 2008," ODF Alliance, 2008; www.odfalliance.org/press/Release20081222-annual-report-odf-2008.pdf.
2. R. Shaw, "Stupid FEMA Tech Blunder Will Thwart Some Katrina Claim Applications," *Huffington Post*, 2 Sept. 2005; www.huffingtonpost.com/russell-shaw/stupid-fema-tech-blunder-_b_7055.html.
3. Y. Savourel, J. Kosek, and R. Ishida, "Best Practices for XML Internationalization," W3C, 2008; www.w3.org/TR/xml-i18n-bp.
4. P. Korn and R. Schwerdtfeger, "Open Document Format v1.1 Accessibility Guidelines Version 1.0," OASIS, 2008; http://docs.oasis-open.org/office/office-accessibility/v1.0/cs01/ODF_Accessibility_Guidelines-v1.0.pdf.
5. OpenDocument Format Alliance, "ODF Annual Report 2008," 2008; www.odfalliance.org/resources/Annual-Report-ODF-2008.pdf.

Rob Weir is a software architect for IBM. His research interests are ODF and personal productivity applications. Weir has a BA in astronomy and astrophysics from Harvard College. He is cochair of the OASIS ODF TC, as well as a member of the OASIS ODF Adoption TC, the OASIS ODF Interoperability and Conformance TC, and a representative in ISO/IEC JTC 1 SC34 for the US. Weir maintains a blog covering ODF and related topics at www.robweir.com/blog. Contact him at robert_weir@us.ibm.com.