OpenDocument and the office experience

WHAT'S INSIDE?

OpenDocument format offers a new approach to data storage and document exchange for office applications. But what does ODF mean for the user? What's inside an ODF file? How portable is ODF? We examine these questions in this month's cover story. **BY DMITRI POPOV**

The process of adopting Open Document Format (ODF) as an open format for office applications in the Commonwealth of Massachusetts was nothing short of a political thriller. It is hard to believe a humble document format could cause so much controversy and political intrigue. OpenDocument, however, is more than just a document format.

ODF represents a new approach to the problem of data exchange for desktop applications, and it heralds the end of the era in which software vendors could exert control over the office tool market by maintaining exclusive knowledge of the formats for file storage.

The real goal, though, is not to defeat Microsoft or promote OpenOffice.org, and the subject of the debate is not whether the Word format is technically superior to that of the OpenOffice Writer format. The ultimate goal is to prevent information loss in our society, which is what often occurs because of proprietary formats. This goal can only be achieved through the transition to open standards, such as OpenDocument Format.

OpenDocument Format is a standard developed by the Organization for the Advancement of Structured Information Standards (OASIS). The original purpose of OpenDocument Format was to standardize the use of XML as a format for storing and exchanging data among office applications. The appearance of OpenDocument Format has already forced closed-source vendors, like Microsoft, to provide more visible alternatives for data storage formats.

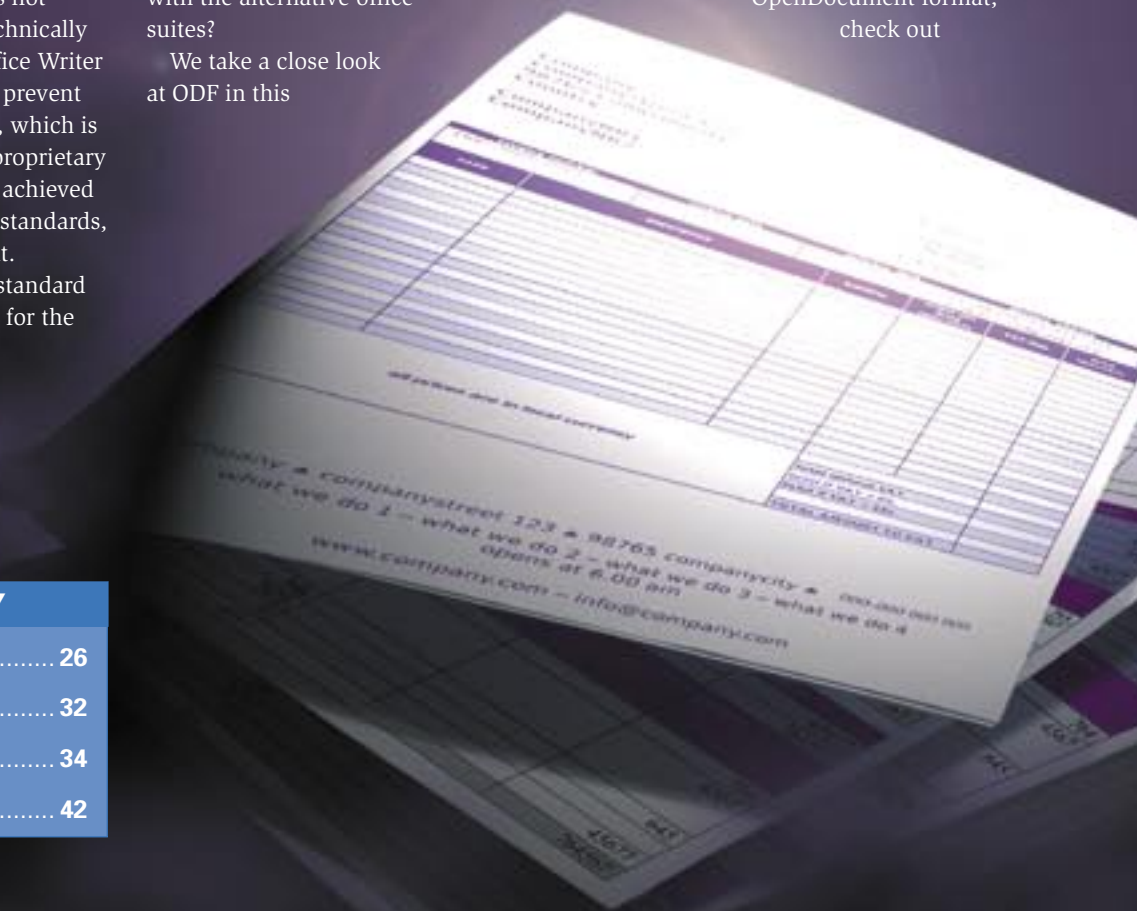
However, despite all the recent attention ODF has received in the news, and despite the fact that users around the world have already started using the OpenDocument format in OpenOffice 2.0 for their daily work needs, little has been written about the ODF format itself. What is ODF? And just how compatible is OpenDocument Format with the alternative office suites?

We take a close look at ODF in this month's Linux Magazine cover story. In this article, we will show you how to get inside an OpenDocument Format document, and we'll show you how the information inside is organized.

In later articles, we'll test the portability of an ODF document, and we'll describe how you can convert an ODF file so it can then be opened in the Microsoft Word program.

Continuing with this office theme, we'll also take a look at some of the popular alternatives to OpenOffice, and we'll examine the recent phenomenon of web-based office applications.

We hope you enjoy this look at OpenDocument format and Linux office alternatives. For some more in-depth information on the anatomy of the OpenDocument format, check out



COVER STORY

ODF Compatibility 26
ODF Add-in 32
Office Suites 34
Web-based Office Tools 42

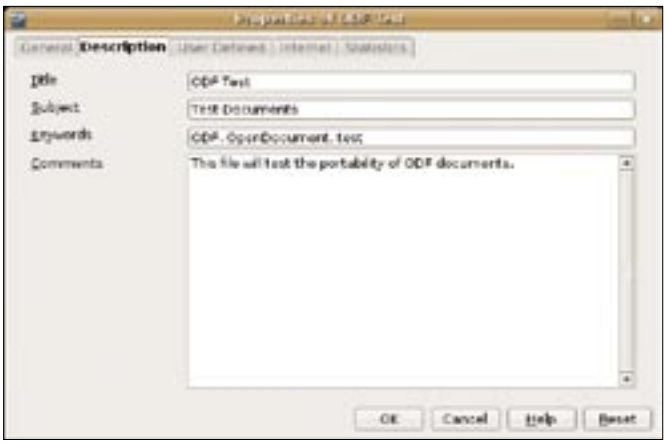


Figure 1: Much of the information stored in meta.xml is visible through the Properties dialog box in OpenOffice 2.0.

OpenDocument Essentials by J. David Eisenberg, a PDF-based book released under the GNU Free documentation license and available at [2].

What's Inside

Office documents include a whole lot more than simple text and numbers. A typical document also contains style and formatting information, in addition to data on the user who created the file, and other bits of information that are useful to the application.

In fact, what we think of as an ODF file is not just a file. An OpenDocument Format file is actually a collection of files that are saved in a Java Archive (JAR), which is a form of compressed zip file. To look inside an OpenDocument Format document, all you need to do is change its *.odt* extension to *.zip* and then just unpack it. The unzipped document will typically contain at least the following files:

- content.xml
- META-INF/manifest.xml
- meta.xml
- mimetype
- settings.xml
- styles.xml

The exact contents of the document can vary. For example, if the document contains images and macros, these elements are included within separate folders.

The Files

Each of the files within an ODF document has a specific purpose. *content.xml* is the key file that contains the actual text content of the document. If you open the *content.xml* file in a text editor, you will notice that the XML markup used in the file has a lot of similarities with HTML.

Even if you are not proficient in XML, you can easily identify some basic markup tags. In Listing 1, for instance, the "Lorem ipsum" string is formatted as Heading 1 and the "Proin velit" sentence is marked as bold.

Like an HTML file, *content.xml* includes text as well as the formatting instructions. The formatting commands associate text with font and style information. In more elaborate documents,

Listing 1: content.xml

```
01 <text:h text:style-name="Heading_20_1"text:outline-level="1">Lorem ipsum</text:h>
02 <text:p text:style-name="Standard"/><text:p text:style-name="Standard">Lorem ipsum dolor sit amet.
03 <text:span text:style-name="T1">Proin velit.</text:span>
```

Listing 2: manifest.xml

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <manifest:manifest xmlns:manifest="urn:oasis:names:tc:opendocument:xmlns:manifest:1.0">
03  <manifest:file-entry manifest:media-type="application/vnd.oasis.opendocument.text" manifest:
   full-path="/"/>
04  <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="content.xml"/>
05  <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="styles.xml"/>
06  <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="meta.xml"/>
07  <manifest:file-entry manifest:media-type="text/xml" manifest:full-path="settings.xml"/>
08 </manifest:manifest>
```

Listing 3: meta.xml

```
01 <meta:generator>OpenOffice.org/2.0$Unix OpenOffice.org_project/680m5$Build-9073</meta:generator>
02 <meta:initial-creator>Dmitri Popov</meta:initial-creator>
03 <meta:creation-date>2006-11-30T11:47:40</meta:creation-date>
04 <dc:creator>Dmitri Popov</dc:creator>
05 <dc:date>2006-11-30T11:58:51</dc:date>
06 <dc:language>en</dc:language>
```

the tags might specify where to place boxes or images within the file.

The rest of the files inside the archive are either auxiliary content files (such as images files embedded in the document) or files that provide some background information. The exact number of files can vary depending on the document and the

What's an Open Format?

Although the industry has no official list of criteria for an "open format," several important considerations come to mind.

The first feature of an open format is open, collaborative development of the specification. ODF, like any other open format, is the result of many discussions and a long development process that involved many parties. These companies and organizations are members of OASIS, an organization that coordinates the development and approval of open standards in general and ODF in particular. While OASIS members participate directly in creating ODF specifications, anyone can follow the development and get access to the relevant information. OASIS is not an exclusive club: anyone can get involved, or at least, keep up-to-date with its work. If you want to join the force, you could start with the OASIS OpenDocument Technical Committee [7], OASIS ODF Adoption Committee [8], or OpenDocument XML.org [9].

A truly open standard must be free of patent or licensing restrictions. This requirement might sound pretty obvious, but it is one of the most important requirements of an open format. This ensures that the developer doesn't have to obtain a license or pay royalties to a third party for using the format. And as a user, you can use the format to store your documents without worrying about being sued for infringing somebody's obscure patent or violating licensing terms. Of course, some vendors or organizations use the term "open" for standards that other vendors or organizations might not consider open. The complex problem of license and patent restrictions is a matter of constant debate throughout the software industry.

An open format must be free of proprietary dependencies and single-vendor functionality. ODF does not rely on any proprietary technologies, and it does not allow a company to add some functionality to it at its convenience. This, among other things, prevents the practice exemplified by Microsoft's attempt to add some proprietary extensions to Java, thus making their version incompatible with the rest of the world.

As an open format, ODF also ensures easy implementation by software developers in their own software as well as interoperability with any application that supports ODF. Since all the specifications in ODF are available and well documented, it should be fairly easy for any developer to add ODF support to the existing application or build new software that uses ODF. For a comprehensive list of software that supports ODF, see the related Wikipedia article [10] and the Supported Software page at the OpenDocument Fellowship's website [11].

With the release of Office 2007, Microsoft introduced a new XML-based document format. For an Open Source perspective on the differences between OpenDocument Format and Microsoft's XML formats, see "Format Comparison Between ODF and MS XML" [12]. Wikipedia provides a summary of this article for those who are looking for a quick recap [13].

If you want to better understand the different aspects of defining open standards, listen to the interview with Danese Cooper at Open Source Conversations [14].

application that saved it. A few files, however, are especially important.

The *manifest.xml* file contains a directory of all the files in the document (see Listing 2). You'll find the *manifest.xml* file inside the *META-INF* directory.

The *meta.xml* file contains the document's metadata, including the application used to create the document, the author's name, the creation date, the language, and the word count (Listing 3). The data in *meta.xml* will be familiar to users who have examined the Document Properties dialog in OpenOffice.org (Figure 1).

Various miscellaneous settings for the document are stored in *settings.xml*. *settings.xml* includes parameters such as Field Auto Update, Save Version on Close, Printer Name, and so forth.

According to the specification, *settings.xml* is intended for storing information that is used by the application that actually created the file. However, these settings may not be supported by other applications that later open the ODF document.

Most modern word processing programs provide the ability to associate a block of text or a paragraph with a style (Figure 2). The ODF *styles.xml* file contains style information associated with the document.

Styles offer a means of separating the formatting details from the content, which keeps the *content.xml* file simpler and more manageable.

The mimetype file contains the MIME type of the document. Some of the possible MIME types for OpenDocument Format files are shown Table 1.

In our case, the MIME type for a text file is *application/vnd.oasis.opendocument.text*.

Automating Open Document

XML is designed for easy automation, and the sheer simplicity and readability of OpenDocument Format allows developers to quickly build tools that support ODF. Of course, the number of possible settings and options within an ODF file is nearly endless. But unless you are writing your own office application from scratch, you won't need to automate the full spectrum of style and configuration options.

A quick and simple strategy for writing a program that builds an ODF file is to create a document that has the format and features of the file you would like the program to generate, then unpack the ODF file (as described earlier in this article) and examine the XML.

You can experiment with changing settings within the XML files, and reload the OpenDocument Format file to see the effects of your changes.

Then, as J. David Eisenberg writes in *OASIS OpenDocument Essentials*, "Once you know how a feature works, don't hesitate to copy and paste the XML from the OpenDocument file into your program. In other words, cheat. It worked

Table 1: OpenDocument File Types

Extension	Description	MIME Type
odt	text document	application/vnd.oasis.opendocument.text
ott	text doc used as template	application/vnd.oasis.opendocument.text-template
oth	text doc used as template for HTML	application/vnd.oasis.opendocument.text-web
odm	global text document	application/vnd.oasis.opendocument.text-master
odg	drawing document	application/vnd.oasis.opendocument.graphics
otg	drawing doc used as template	application/vnd.oasis.opendocument.graphics-template
odp	presentation document	application/vnd.oasis.opendocument.presentations
otp	presentation doc used as template	application/vnd.oasis.opendocument.presentations-template
ods	spreadsheet document	application/vnd.oasis.opendocument.spreadsheet
ots	spreadsheet document used as template	application/vnd.oasis.opendocument.spreadsheet-template
odc	chart document	application/vnd.oasis.opendocument.chart
otc	chart doc used as template	application/vnd.oasis.opendocument.chart-template
odi	image document	application/vnd.oasis.opendocument.image
oti	image doc used as template	application/vnd.oasis.opendocument.image-template
odf	formula document	application/vnd.oasis.opendocument.formula
otf	formula doc used as template	application/vnd.oasis.opendocument.formula-template

INFO

[1] OASIS: *http://www.oasis-open.org/*

[2] *OASIS OpenDocument Essentials* by J. David Eisenberg: *http://books.evc-cit.info/*

[3] odt2txt: *http://www.freewisdom.org/projects/python-markdown/odt2txt.php*

[4] ODF concerns: *http://blogs.zdnet.com/Ou/?p=101*

[5] More ODF concerns: *http://blogs.zdnet.com/Ou/?p=119*

[6] "Why OpenDocument Won (and Microsoft Open XML Didn't)": *http://www.dwheeler.com/essays/why-opendocument-won.html*

[7] OpenDocument Technical Committee: *http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=office*

[8] OASIS ODF Adoption Committee: *http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=odf-adoption*

[9] OpenDocumentXML.org: *http://opendocument.xml.org/*

[10] Wikipedia on OpenDocument support: *http://en.wikipedia.org/wiki/OpenDocument_software*

[11] OpenDocument Supported Software: *http://opendocumentfellowship.org/applications*

[12] "Format comparison between ODF and MS XML": *http://www.groklaw.net/articlebasic.php?story=20051125144611543*

[13] Wikipedia ODF and MS XML comparison: *http://en.wikipedia.org/wiki/Comparison_of_OpenDocument_and_Microsoft_Office_Open_XML_formats*

[14] Danese Cooper on open standards: *http://osc.gigavox.com/shows/detail1222.html*

for me when I was writing this book, and it can work for you" [2].

XML is much easier to rescue and repair than a binary format. Even if the application itself fails to recover the XML-based file, you can still get your data out of it directly because, unlike binary files, XML-based files are stored in a human-readable form.

A good example of such a simple yet effective tool for converting and rescuing ODF text files is the odt2txt Python script [3]. odt2txt converts OpenOffice.org Writer documents into plain text files, turning the document's formatting into Markdown syntax.

Final Word

Although ODF offers a viable and open alternative to proprietary formats, it does have some weak points. For example, some concerns were raised regarding the format's efficiency with large amounts of data (see [4] and [5]).

As dependency on electronic document storage and exchange in our society grows, so does the concern of interoperability issues and information loss due to proprietary formats.

OpenDocument Format offers a solution to these problems, and it's really only a matter of time before governments and businesses realize the importance of storing data in open formats such as ODF [6]. ■

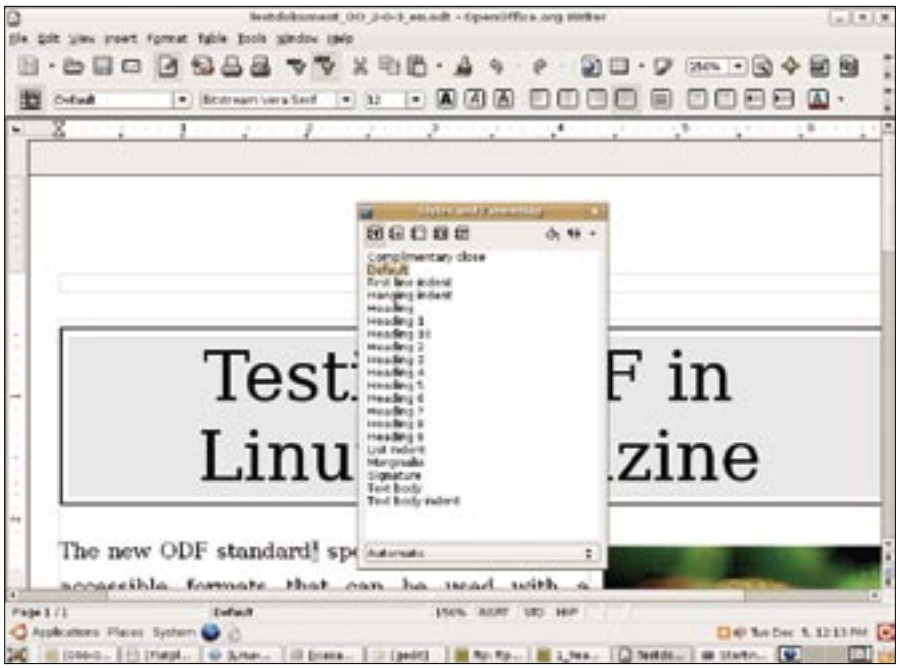


Figure 2: Most word processors let the user format a document using predefined styles.