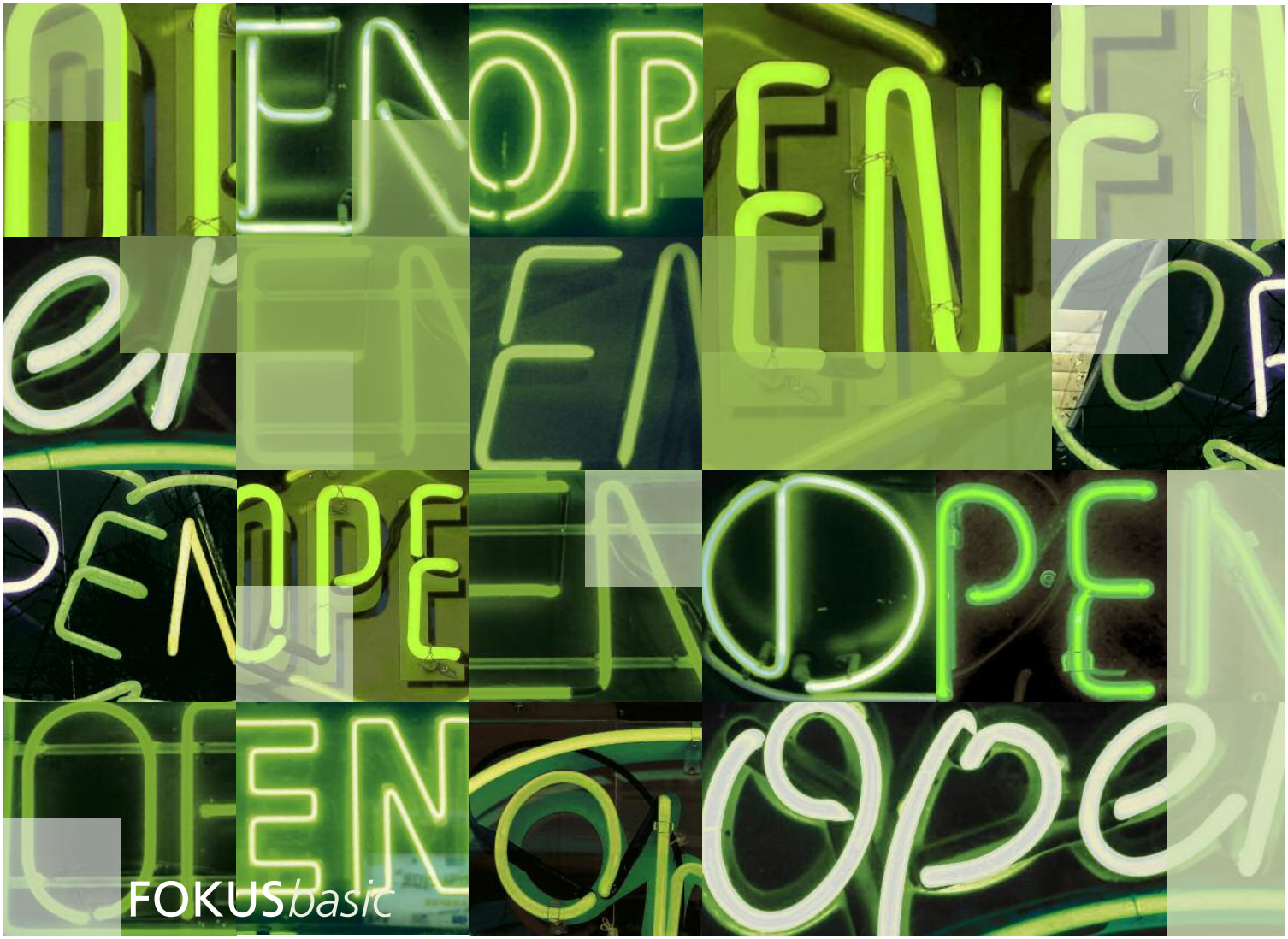
Fraunhofer FOKUS

Document Interoperability

Open Document Format and Office Open XML

Dr. Klaus-Peter Eckert · Jan Henrik Ziesing · Ucheoma Ishionwu



FOKUS*basic*

Imprint

Publisher

Fraunhofer-Institute for
Open Communication Systems FOKUS
Kaiserin-Augusta-Allee 31
10589 Berlin, Germany

Contact

Competence Center Elan:
Electronic Government and Applications
Telephone +49 (0)30 3463-7115
eMail elankontakt@fokus.fraunhofer.de
www.fokus.fraunhofer.de/egov-lab

Authors

Dr. Klaus-Peter Eckert
Telephone +49 (0)30 3463-7227
eMail klaus-peter.eckert@fokus.fraunhofer.de

Jan Ziesing
Telephone +49 (0)30 3463-7312
eMail jan.ziesing@fokus.fraunhofer.de

Ucheoma Ishionwu
Telephone +49 (0)30 3463-9225
eMail ucheoma.ishionwu@fokus.fraunhofer.de

Printing and Bindery

Mediendienstleistungen des
Fraunhofer-Informationszentrum Raum und Bau IRB, Stuttgart

Printed on acid-free and chlorine-free bleached paper.

Fraunhofer Information-Centre for Regional Planning and Building Construction IRB
P.O. Box 80 04 69, D-70504 Stuttgart
Nobelstrasse 12, D-70569 Stuttgart
Fon +49 (0) 7 11/9 70-25 00
Fax +49 (0) 7 11/9 70-25 08
E-Mail verlag@fraunhofer.de
URL http://verlag.fraunhofer.de

Acknowledgements

The authors especially wish to thank the experts on document formats and applications Dirk Vollmar & Wolfgang Keber from DIaLOGIKa as well as Mario Wendt from Microsoft, Mohamed Zergaoui from Innovimax and Florian Reuter from Novell. They helped to develop important basics and ideas and supplied valuable comments on prior versions of this White Paper. Special thanks are also due to the DIN and ISO which picked up on some of the ideas prior to this White Paper and are currently developing them further as part of the standardization process.

Contents

1 Introduction

OASIS Open Document Format ODF 1.0 (ISO/IEC 26300) and Office Open XML (ISO/IEC 29500) are both open document formats for saving and exchanging word processing documents, spreadsheets and presentations. Both formats are XML based but differ in design and scope.

OASIS ODF 1.0 was published by OASIS in May 2005 and accepted as an international standard by ISO (ISO/IEC 26300) in December 2006. Office Open XML was first approved in December 2006 by the ECMA International General Assembly as ECMA-376. An updated version was published in November 2008 by ISO (ISO/IEC 29500). The corresponding version, ECMA-376 2nd edition, was published in December 2008.

The White Paper "Document Interoperability: Open Document Format and Office Open XML" addresses both technical and strategic decision makers in the public sector. It analyzes how both standards implement the most important document features, and if and how these features can be translated between the two formats. The Paper targets users of both document formats as well as template designers whose competences cut across the spectrum of XML and XML-related technologies which directly or remotely deal with one or both of the two standards. The Paper will be of great assistance to those seeking to exchange documents between formats, to extract data from or import data into documents, or to write applications supporting the two formats.

This White Paper aims at analyzing the two standards and their underlying concepts in terms of interoperability issues for a selected set of features. It analysis the way these features are implemented in both standards and estimates the degree of translatability between them using a table-based comparison. The document serves as a preliminary technical translation guideline for evaluating translatability between certain parts of the two standards. It does not compare different implementations which can cause additional kinds of interoperability problems.

Both Office Open XML and Open Document formats are basically descriptions of schemas used for word processing documents, spreadsheets and presentations created by office application suites. Both are open formats. A key design objective is to guarantee long term access to data without the legal or technical barriers associated with proprietary binary formats. XML schema definitions are normative parts of both standards.

The easiest and most flexible way of manipulating documents is to separate a document's layout from its content. Editing the layout and data components independently of one another affords considerable flexibility in creating and editing office documents. Defining the structure and content of documents has been the focus of both standards. A document's layout is ultimately governed by the implementation of the office suite, in particular by the rendering engine. Thus, using exactly the same standard to describe a document does not guarantee that different office suites will produce identical layouts. Consequently this White Paper focuses more on the definition of guidelines for the translation of document structure and content than on the preservation of document layout.

In this White Paper the two standards will be examined in their universality and not by comparing specific implementations such as Microsoft Office or OpenOffice. For this reason, various examples have been developed using a simple XML editor which supports both standards. The names of

specific implementations may be used in the use cases to illustrate the real world scenario behind the use case. The figures in this White Paper are created for illustration purposes, using available tools such as OpenOffice 3.1 and Microsoft Office 2007 SP2. It should not be assumed that the current versions of these implementations support all the features needed to implement the use case, especially the document standards and the translation between them.

Several use cases do not mention existing tools, but rather use abstract names such as document format A (DF-A) and document format B (DF-B).

The White Paper begins with a short overview of XML based document standards. It presents typical use cases characterizing scenarios where specific features supported by both document formats are used. It then analyzes the most important features of one document format and show how those features can best be represented in the other format. The White Paper then reviews the concepts, architectures and various features of the two document formats in order to provide a good understanding of the formats' common features and especially their differences. Most features can be translated to the other format with varying degrees of fidelity. For each feature, we provide detailed information on the extent to which that feature can be translated.

The following abbreviations will be used throughout this White Paper:

- *ODF*, which stands for Open Document Format (ISO/IEC 26300:2006).
- *OOXML*, which stands for Office Open XML (ISO/IEC 29500:2008).

We hope that this White Paper will be useful in understanding how the ODF and OOXML standards compare, and how their functionality can be mapped between the two formats.

2 XML based Document File Formats

In the early years of the personal computer, different office applications each used their own proprietary binary file formats. Binary file formats convert human-readable content into machine-readable representations in binary form. Proprietary formats closely connect the file format to the application producing it.[1]

The first free and open standard for a document file and interchange format was the *Open Document Architecture and Interchange Format* (ODA/ODIF) published by ISO between 1989 and 1999 as ISO 8613-1:1989, but which failed to find broad acceptance. The *Open Document Architecture* now has no market relevance, but its ideas and concepts have indeed influenced standards for document file formats broadly used today.[2]

Another important early standard influencing development of XML-based document file formats was ISO 8879:1986 - the *Standard Generalized Markup Language* (SGML).[3] SGML is a meta-language for defining markup languages for documents. SGML was originally designed to enable sharing of large machine-readable documents which have to remain readable for decades. SGML is the substantially more comprehensive and powerful predecessor of the *eXtensible Markup Language* (XML) designed for ease of implementation. XML is now a W3C open standard widely used by numerous applications.[4]

The original Sun specification for Open Document Format, adopted by OASIS in 2005 as the OASIS ODF 1.0 standard, was developed between 2000 and 2002 with the following objective:

"To create as a community, the leading international office suite that will run on all major platforms and provide access to all functionality and data through open-component based APIs and an XML-based file format." [5]

Microsoft followed suit in 2006 via the *Open Specification Promise* (OSP)[6] by opening the format of its 2007 version of the Microsoft office suite (version 12) for which it also uses XML as an exchange and storage format. This format is published as ECMA-376 1st edition.

2.1 eXtensible Markup Language

The eXtensible Markup Language (XML) is a deliberately simple and straightforward text format for the exchange and storage of data. The core of the XML format is the coupling of data with its corresponding mark-up (in the form <start-tag> data <end-tag>; for example: <real estate price> 220.000 </real estate price>. The tags, which are always in angle brackets, enable human readers to understand the meaning of the data, and computer systems to process the data – in the example given, the figures for the real estate price can be clearly identified. The ease and straightforwardness

[1] (Ditch, 2007)
[2] (ISO, 1989)
[3] (ISO, 1986)
[4] (W3C, 2006)
[5] (OpenOffice, 2002)
[6] (Microsoft, 2006)

of XML means that it now enjoys widespread acceptance, and has become near-pervasive. XML is one of the major forms of data exchange in all eCommerce and eGovernment sectors.[7]

2.2 Office Open XML

Office Open XML is a file format originally developed by Microsoft as a successor to its earlier Office 2003 file formats. Office Open XML is used for representing spreadsheet, presentation and word processing documents. In 2006 Office Open XML became an ECMA standard (ECMA-376). In 2008, a revised version of ECMA-376 became an open ISO standard (ISO/IEC 29500:2008). The ISO standard, which has its equivalent in the ECMA-376 Second Edition, freely available in public domain, will be supported in Microsoft's Office 2010 (code named Office14).[8]

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[9] 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** (129 pages)
   Part 2 of the ISO29500 standard[10] defines the Open Packaging Conventions (package model, physical package) along with the core properties, thumbnails and digital signatures.

3. **Markup Compatibility and Extensibility** (40 pages)
   Part 3 of the ISO29500 standard[11] clearly specifies how elements and attributes should be introduced by future versions or extensions of Office Open XML documents. It describes extension facilities of OOXML documents while also providing a method by which consumers can obtain a baseline version of the OOXML document (a version without extensions) for interoperability.

4. **Transitional Migration Features** (1465 pages)
   Part 4 of the ISO29500 standard[12] 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.

The standard specifies six levels of document and application conformance, *strict* and *transitional* for WordprocessingML, PresentationML and SpreadsheetML respectively, together with corresponding

[7] (Schmidt, et al., 2006)
[8] (Microsoft, 2008)
[9] (ECMA-376-1, 2008)
[10] (ECMA-376-2, 2008)
[11] (ECMA-376-3, 2008)
[12] (ECMA-376-4, 2008)

schema definitions. Transitional conformance allows inclusion of XML attributes used to provide compatibility with older versions of Office documents. Details are specified in Part 4 of the standard. Strict conformance restricts the number of XML attributes to the core defined in Part 1. Microsoft Office 2010 produces transitional conformant documents.

The standard also specifies application descriptions of the types *base* and *full*. Base applications are obliged to understand at least one feature of their conformance class, while full applications must support all features. The introduction of domain-specific document categories and supporting applications can be expected in the future.

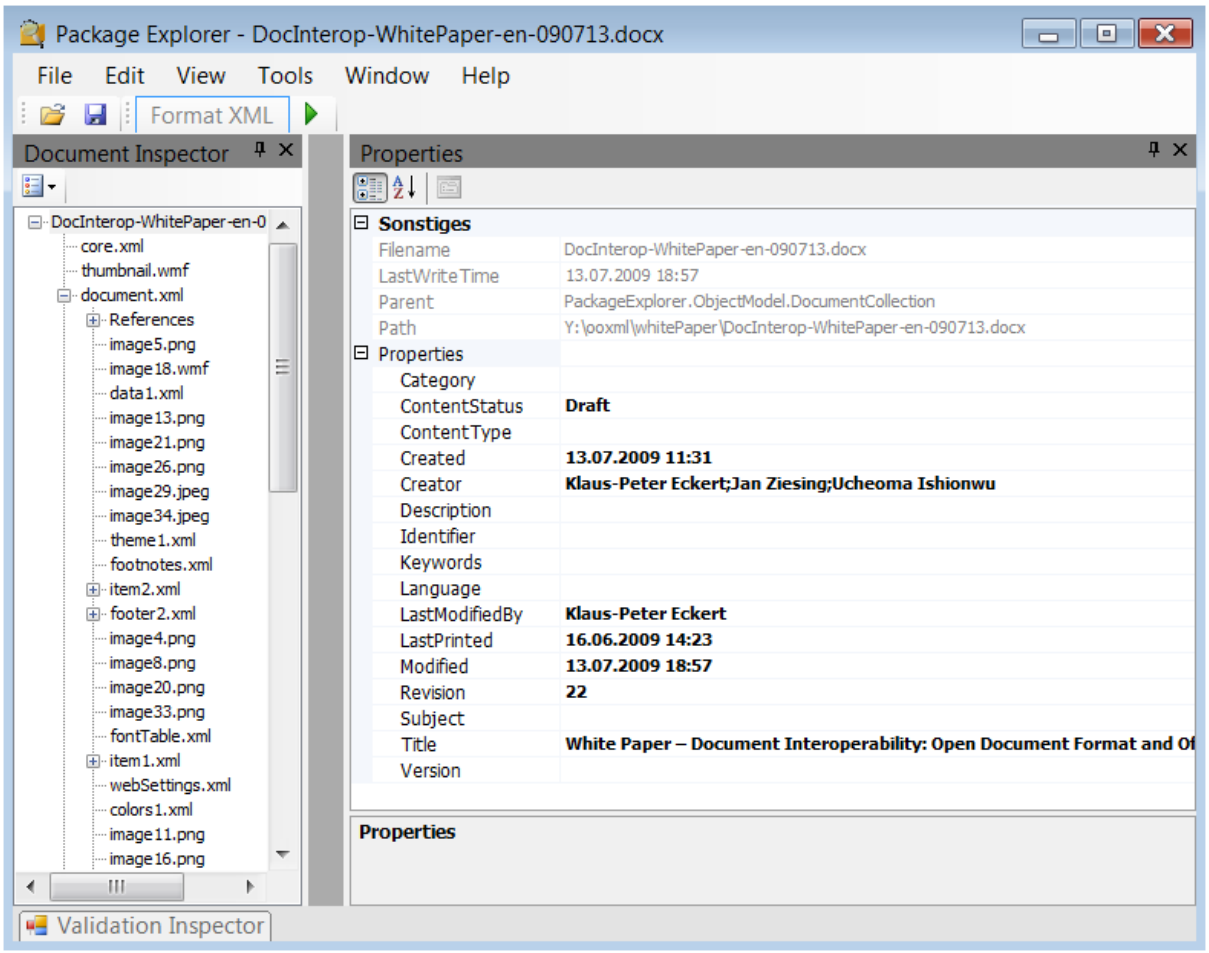


Figure 1: This White Paper opened in Package Explorer

An Office Open XML document file is actually a compressed zip package containing mainly XML-based files. The Office Open XML file can be extracted using different tools. Examples include Package Explorer[13], XMLSpy[14], and the Oxygen XML Editor[15]. As an example this White Paper is opened in the Package Explorer Application as shown in Figure 1.

[13] (Vugt, 2009)
[14] (Altova)
[15] (Oxygen)

The structure of the ZIP container is defined via the Open Packaging Conventions (OPC), an abstraction layer between the physical file / directory layout inside the ZIP file, and the document structure. OPC defines the concepts of *Parts* containing data and *Relationships* connecting the Parts.

At the root lies the so-called "Content Type Stream" which identifies the overall document's type as well as the content type of its individual *Parts*. The root relationship defines the location in the ZIP file of the main document *Part*. Depending on the document type and the contents of the document, the main *Part* will be connected to further *Parts* and/or external documents via *Relationships*.

Office Open XML contains specifications for the following three document types:

- Word processing documents
- Spreadsheets
- Presentations

The following diagram illustrates the relationships between the technologies upon which Office Open XML is based:

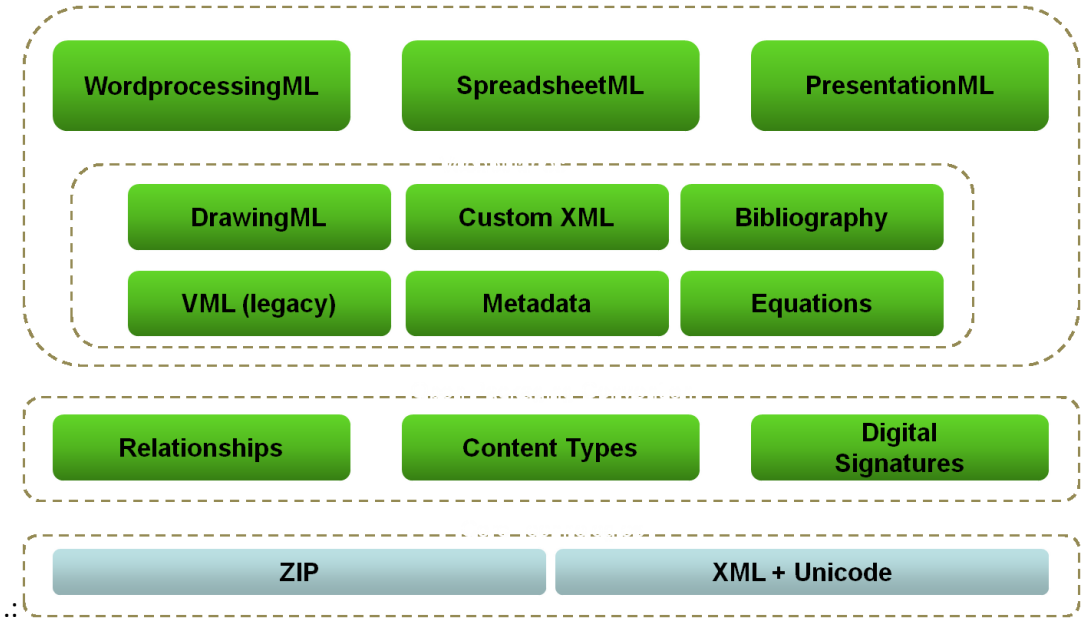


Figure 2: OOXML architecture[16]

Each document type is defined via its own markup language and uses shared languages for functionalities common to all three document types (e.g. drawings, metadata, etc.).

2.3 Open Document Format

OpenDocument was originally developed by Sun Microsystems beginning in 2000, as the XML-based format for StarOffice and OpenOffice. In 2002, the standardization process was initiated at OASIS in the newly created *OASIS Open Office XML Format Technical Committee*. This TC was renamed to *OASIS Open Document Format for Office Applications TC* in January 2005, and in May 2005 the

[16] Graphic by Microsoft (Microsoft, 2007)

standard was published as *OASIS Open Document Format for Office Applications*, abbreviated as *OpenDocument* or *ODF*[17]. In 2006, *Open Document Format for Office Applications v.1.0* became an ISO Standard [ISO/IEC 26300]. *Open Document Format for Office Applications v.1.1*[18] is the latest version, standardized and published by OASIS. At the time of writing (July 2009) Version 1.2 is still in a drafting stage. While version 1.0 of the ODF standard only consists of one part, the current working draft (version 1.2)[19] is structured into three parts: core, formulas, and packages.

The ISO/IEC 26300:2006(E) document is structured as follows:

- Chapter 1: Introduction to the OpenDocument format
- Chapter 2: Document structure
- Chapter 3: Meta-information
- Chapter 4: Text
- Chapter 5: Paragraph content
- Chapter 6: Text fields
- Chapter 7: Text indices
- Chapter 8: Table content
- Chapter 9: Graphical content
- Chapter 10: Chart content
- Chapter 11: Form content
- Chapter 12: Content common to all documents
- Chapter 13: Integration of SMIL animation markup into the OpenDocument schema.
- Chapter 14: Style information content
- Chapter 15: Formatting properties used within styles
- Chapter 16: Data types
- Chapter 17: Packages [20]

To date (July 2009) most applications use ODF version 1.1, or even drafts of version 1.2. Examples of currently available implementations of ODF include OpenOffice.org, StarOffice, NeoOffice, KOffice, Google Docs, Lotus Symphony, Apple TextEdit, as well as Microsoft Office 2007 Service Pack 2 and the Windows 7 implementation of Wordpad 2009.

Documents using ODF have an internal structure similar to that of Java-archive files (JAR-files). Like the JAR-file they contain a manifest file which declares the type and location of the files contained in the archive.

Unlike OOXML, ODF contains no abstraction layer above the physical file layout within the ZIP archive; instead, fixed file names are used for document content (content.xml), style information (styles.xml), meta information (meta.xml) and application settings (settings.xml). These files are placed in the root directory of the archive; the manifest file (manifest.xml) resides in the subdirectory

[17] (OASIS, 2005)
[18] (OASIS, 2007)
[19] (OASIS, 2009)
[20] (ISO, 2006)

META-INF. For unencrypted documents, the ODF standard mandates the presence of a thumbnail representation of the document, in PNG format, at the location Thumbnails/thumbnail.png.

In order to allow easier content-type recognition by operating systems and processing applications, ODF documents may contain a file “mime type” containing nothing but the mime type in the root directory. If this file is present, the standard specifies that this type must not be compressed and must be the first entry in the ZIP archive so that the information it contains is located at a fixed offset within the binary representation of the document.

Figure 3 illustrates the minimal structure of an ODF document stored in a ZIP container:

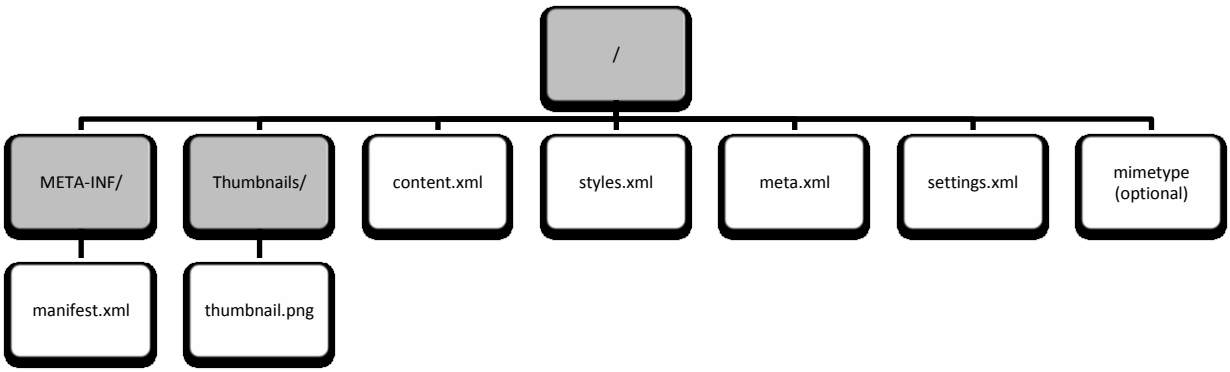


Figure 3: Minimal ODF structure

3 Basic Principles

3.1 Structure of the White Paper

3.1.1 Use case template

To facilitate comparisons and a quick overview, use cases are described using the following template:

Description:

- *Describes the scenario/story the use case is going to tell*
- *Includes one or more figures demonstrating the use case (optional)*
- *Defines the translation type and fidelity to be demonstrated*

Implementation:

- *Describes the features necessary to implement the use case*

<table>
<tr><td colspan="3">Use case name:</td></tr>
<tr><td colspan="3">Translation type and fidelity</td></tr>
<tr><td rowspan="9"></td><td>One-trip translation</td><td>☒☐</td></tr>
<tr><td>Round-trip translation</td><td>☒☐</td></tr>
<tr><td>Presentation instructions</td><td>☒☐</td></tr>
<tr><td>Document content</td><td>☒☐</td></tr>
<tr><td>Dynamic content</td><td>☒☐</td></tr>
<tr><td>Metadata</td><td>☒☐</td></tr>
<tr><td>Annotations</td><td>☒☐</td></tr>
<tr><td>Document parts</td><td>☒☐</td></tr>
<tr><td>Additional fidelities if needed</td><td>☒☐</td></tr>
<tr><td colspan="3">Required features:
• Feature a including references to standards
• Feature b including references to standards</td></tr>
</table>

Requirements:

- *Describes the expected behavior of a feature's translation between both standards*

- *Describes how the document(s) used in the use case should be defined to achieve the intended fidelity*

Conclusion:

- *Compare the applicable features in both standards and the translation rules and fidelity as elaborated in section 0.*

3.1.2 Use case scenario

All use cases are defined as parts of an overall scenario describing a typical information-sharing situation between cooperating public authorities and between citizens and public authorities. Assume employee A of a public authority in federal state A and employee B of another public authority in federal state B wish to exchange documents independent of the office suites used by their agencies. Microsoft Office and OpenDocument suites are all in broad use for generating documents, with millions of documents extant in legacy and actual formats.

For purposes of the use case scenario, employee A uses Microsoft Office 2003, and employee B uses OpenOffice. Both public authorities use a central forms server with a built-in template catalog. In addition to files using the storage format of their respective office suites, both public authorities can create pdf-files.

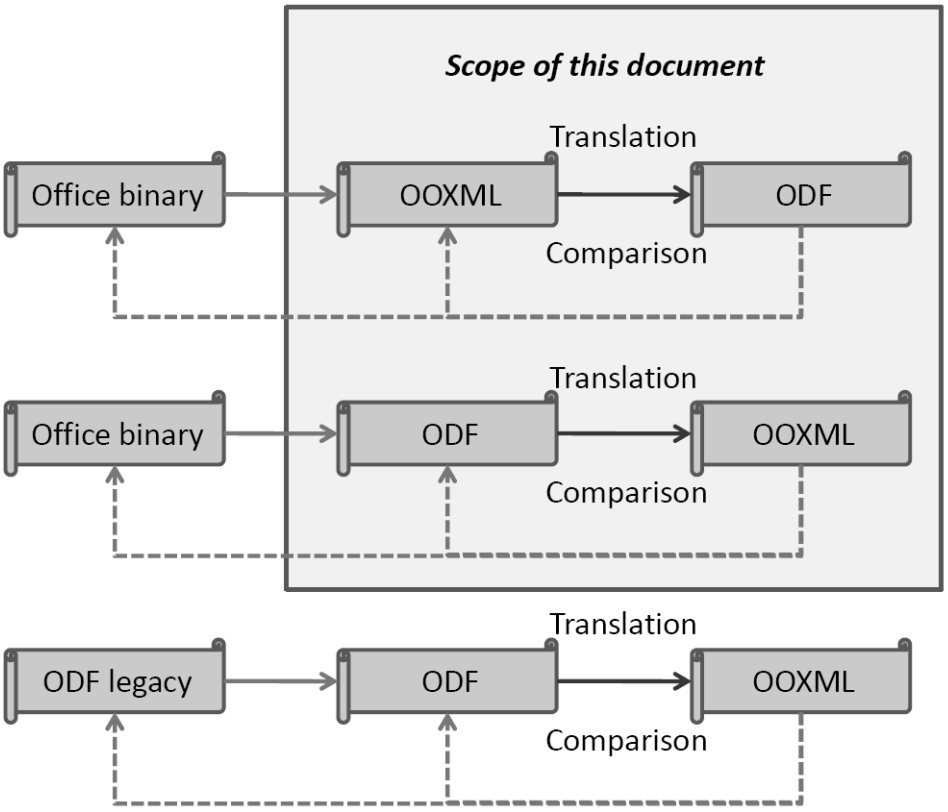


Figure 4: Overall use case scenario

As both pubic authorities see the advantage of sharing XML documents, they decide to test both available XML formats to check how readily documents can be translated between both standards.

1. Employee A sends employee B a migrated doc/ppt/xls file as an OOXML file.
2. Employee B converts employee A's OOXML file into ODF and sends it back to employee A, who compares the files.
3. Employee B sends employee A a migrated doc/ppt/xls file as an ODF file.
4. Employee A converts employee B's ODF file into OOXML and sends it back to employee B, who compares the files.
5. Employee B converts an OpenDoc legacy file to ODF and sends it to employee A.
6. Employee A converts employee B's ODF file into OOXML and sends it back to employee B, who compares the files.

The intention of this paper is not to review the technical infrastructure or applications, but rather to focus on the documents the two employees wish to share with each other, and the capabilities and restrictions of both XML-based, standardized formats based on their actual specifications. Hence we will focus on different document features and elaborate on their compatibility and translatability in corresponding use cases. Such a focus on translatability between the standards merely addresses a subset of the whole challenge of document translation between public authorities and citizens.

3.2 Approach

This White Paper takes a use-case-based approach to identify the requirements to be considered for translation between ODF and OOXML. As depicted in Figure 5, use cases are selected and categorized along two lines: *type of translation* and *fidelity of translation*. This approach covers all aspects of translation between the two document formats. Both standards define a storage and exchange format for documents, including information about both a document's presentation and its content. Presentation of documents itself is beyond the scope of the actual standards, and thus beyond the scope of both the translation process and this Paper.

Another important category of uses cases, graphic fidelity between different layout engines (i.e. layout implementations), is also beyond the scope of this Paper. In such use cases, different layout engines are provided with the same information, but may produce visually different results. Since the actual process of layout is not described by either the ODF or the OOXML standard, this Paper does not deal with such use cases. However it does cover preservation of layout information around format translations as part of the presentation instructions so that the *selfsame* layout engine can produce the same visual result from the same information encoded in different formats.

Use case descriptions reference section 3.1.2 for the description of demonstrated translation types and fidelities, and section 5 for a comparison of required features and functionalities.

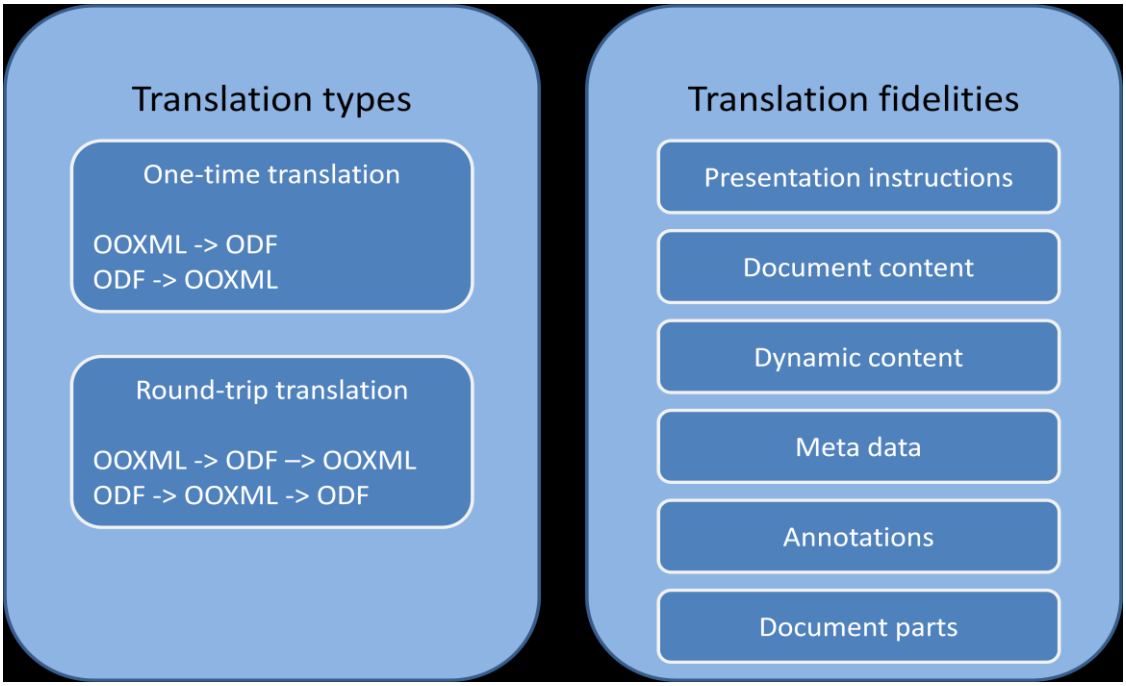


Figure 5: Use case category overview

The definition of translation fidelity considers the following document properties:

- **Presentation instructions** include all layout and presentation related information such as fonts, spacing, margins, and animation in office documents.
- **Document content** (user content) covers all aspects of content defined directly by the user of a document.
- **Dynamic content** covers all aspects of automatically generated content, calculations or form functionalities such as fields, generated tables, or dynamic references.
- **Metadata** cover all information apart from the core document content. Metadata are used to describe meta information about the document such as generator, version, authors, and to ensure the accessibility of documents, for instance by using certificates.
- **Annotation** covers all aspects of annotations to a document including comments, change tracking, and collaborative functions.
- **Document parts** covers all aspects of structural document features such as headlines, tables, listings, tables, or captions.

4 Use Cases

4.1 Word Processing Documents

4.1.1 Empty document

Description:

When a new document is created either in ODF format or in OOXML format, the user sees initially an empty document. When the document is saved without any further editing, a document is generated without user content but with some initial metadata and presentation instructions. This initial content should be preserved as far as possible during the translation process.

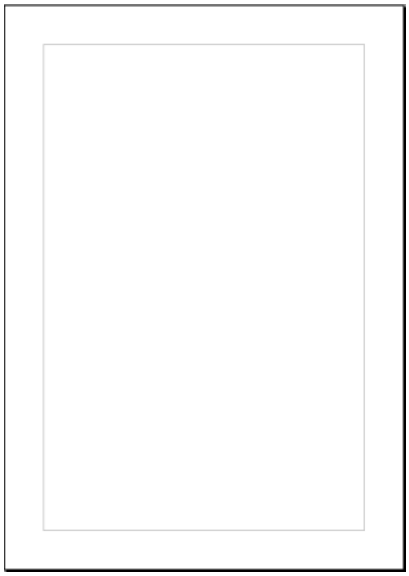


Figure 6: Empty word processing document

Implementation:

Use case name: Empty document		
Translation type and fidelity		
	One-trip translation	☐
	Round-trip translation	☒
	Presentation instructions	☒
	Document content	☐
	Dynamic content	☐
	Metadata	☒

	Annotations	☐
	Document parts	☐

Required features:
• Metadata
  ○ OOXML: section 8.3; 17.*
  ○ ODF: section 3.1		

Requirements:

The following behavior should be obtained no matter which standard is applied:

- Presentation and style instructions remain unchanged
- Metadata remain unchanged

It is not expected that both standards will necessarily use similar defaults for metadata.

Conclusion:

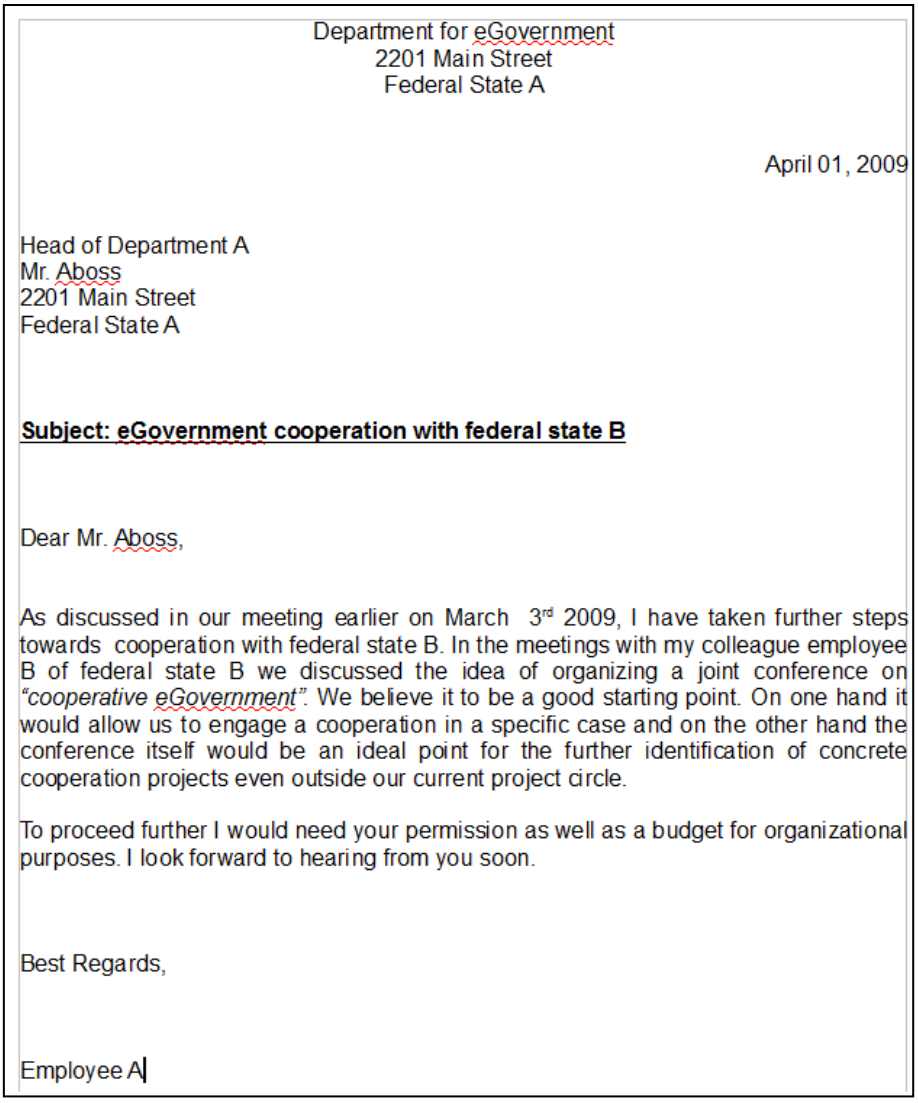
When an empty document defined in the other standard is opened, presentation instructions can be preserved. However, the initial view of the empty document may be slightly different, depending on the rendering engine. Metadata can be translated accordingly, even though some information like the "generator" may be modified.

4.1.2 Simple text formatting

Description:

This use case describes translating a business letter between the ODF and OOXML standards, with a special focus on formal aspects.

The scenario starts with employee A from federal state A, who writes a letter to employee B of federal state B about the possible means of cooperation in eGovernment matters and about the joint organization of a conference on "Cooperative eGovernment". The letter is intended to be sent to both their heads of department. Employee A works on his laptop, using an application which supports format A. The letter looks like the one depicted in Figure 7. After finishing the letter, employee A e-mails it to employee B who imports the document to format B with a tool and adds the name of his head of department, checks for mistakes and sends the document back to employee A. Employee A opens the document again, looks at the now finalized form, saves it in his boss's preferred format A, and forwards it to his departmental head.



Department for eGovernment
2201 Main Street
Federal State A

April 01, 2009

Head of Department A
Mr. Aboss
2201 Main Street
Federal State A

Subject: eGovernment cooperation with federal state B

Dear Mr. Aboss,

As discussed in our meeting earlier on March 3rd 2009, I have taken further steps towards cooperation with federal state B. In the meetings with my colleague employee B of federal state B we discussed the idea of organizing a joint conference on *"cooperative eGovernment"*. We believe it to be a good starting point. On one hand it would allow us to engage a cooperation in a specific case and on the other hand the conference itself would be an ideal point for the further identification of concrete cooperation projects even outside our current project circle.

To proceed further I would need your permission as well as a budget for organizational purposes. I look forward to hearing from you soon.

Best Regards,

Employee A

Figure 7: Sample letter

Implementation:

This sample letter makes use of all typical text formatting features. There is centered text on the top and the date information is positioned on the right. The receiver's address is aligned on the left. The letter's body paragraph is in block format. The text formatting contains a bold paragraph as the subject line, and embedded italic characters in the body text. At the end of this document an image is embedded representing the signature of the author. Layout and structure of the document must be preserved during translation.

Use case name: Simple text formatting		
Translation type and fidelity		
	One-trip translation	☐

	Round-trip translation	☒
	Presentation instructions	☒
	Document content	☒
	Dynamic content	☐
	Metadata	☐
	Annotations	☐
	Document parts	☒

Required features:

- Text formatting
  - OOXML: section 8.3, 17.*
  - ODF: section 2.3, 4, 9.5, 14.*, 15.4
- Paragraph formatting
  - OOXML: section 15.2, 17.*
  - ODF: section 2.8, 4.2, 9.3, 7.12, 14.*, 15.*

Requirements:

This scenario requires the preservation of presentation instructions during multiple translations of a formal business document. A formal business letter is a common example of the application of basic text-processing functionality. A formal letter should strictly conform to a set of guidelines which can be divided into aspects of presentation and content. Regardless of the text-processing applications used to create it, a business letter's appearance and structure should remain identical throughout the translation process.

Conclusion:

The tables "text formatting" and "paragraph formatting" in sections 5.2.1 and 5.2.2 show how far the required features can be translated between the two standards. Simple text formatting such as bold or italic characters, and paragraph formatting such as text alignments can easily be converted between the two formats, with the exception of "theme fonts," which are not supported in ODF.

4.1.3 Documents of public authorities

Description:

Employee A has to fill out a format A based travel application form for his planned journey to the conference on cooperative eGovernment. He sends this to his accounts and human resources department for further processing. The department opens it in format B and should be able to read and further process the application. Figure 8 gives a screenshot example of the travel application.

Travel Application
- Federal State A -
Department for eGovernment

Traveler	Name		Surname	
	Resident in		Working in	
	Travel number			

Travel details	Destination		
	Purpose		
	Time and date		
	Reasons	If flying	
		If using private car	

Budget	Project name	
	Cost unit	
	OE	

Please explain in detail the necessity of the journey:

Employee

Employer

Figure 8: Travel application as an example of a basic document from public authorities

Implementation:

Many different functionalities are used in this document. In the header a graphic is embedded while a text gives information about the document type, its purpose and its application domain. A table is used to structure information (alternatively a form field could be used as shown in use case 4.1.9). A text block enables the editing of free text at a defined location. Furthermore, different text formatting functionalities together with a list are used.

Use case name: Text documents of public authorities		
Translation type and fidelity		
	One-trip translation	☒
	Round-trip translation	☐
	Presentation instructions	☐
	Document content	☒
	Dynamic content	☒
	Metadata	☒

	Annotations	☐
	Document parts	☐
Required features: • Images and vector graphics ○ OOXML: Section 15.2.13 (15.2.14) ○ ODF: Section 9.3.2 • Whitespaces and preserved elements and attributes ○ OOXML: Sections 17.15.1.18, 17.18.7 ○ ODF: Section 1.6 • Text formatting ○ OOXML: Section 17.3.2 ○ ODF: Section 15.4 • Header and footer ○ OOXML: Section 11.3, 17.10 ○ ODF: Section 10.2, 14.3, 15.3 • Tables ○ OOXML: Section 17.4/6; 18.3/8 ○ ODF: Section 8.*; 15.* • Itemization and numeration ○ OOXML: Section 8.3; 17.9 ○ ODF: Section 4.3		

Requirements:

When human resources reviews the document, it should be displayed in exactly the same way as in employee A's application. Images and text field should look the same and retain the same information.

Conclusion:

The document's translation between the standards is difficult from a variety of viewpoints. While the text formatting issues involved in the example are highly translatable (see use case 4.1.2), tables, itemization, numeration and graphics can cause difficulties if they have to be translated (see use cases 4.1.4.; 4.1.5.; 4.1.10.). Even greater problems might arise here due to the different processing of *white spaces* in combination with the *preserve* attribute. Textual header and footers are translatable.

For further details see: 5.2.1, 5.2.4, 5.2.5

4.1.4 Tables and field functions

Description:

After general approval by the two heads of department, employee A plans to give a cost estimate for the joint conference to employee B. After filling out the estimates in a document using format A,

employee A emails the document to employee B. Employee B saves the document in format B and emails it to his colleagues.

Figure 9 shows a brief excerpt from the cost estimate:

Budget estimate for the event on "Cooperative eGovernment" **www.egov-coop.gov**		*Wednesday, April 30th, 2009*	
Organizations & domains		**Forecast 2009**	
		2nd Quarter	*3rd Quarter*
Federal state A	*Internal*	50.000 $	70.000 $
	External	30.000 $	40.000 $
Federal state B	*Internal*	50.000 $	70.000 $
	External	30.000 $	40.000 $

Figure 9: Sample table

Implementation:

One of the more advanced features of text processing is the usage of tables and predefined field functions, as seen in Figure 9. This excerpt shows a table with joined cells and common text formatting. Cells are joined up to span multiple rows and columns. Different cell alignment appears as left, center and right aligned text. A hyperlink is inserted into the header row of the table.

Use case name: Tables and field functions		
Translation type and fidelity		
	One-trip translation	☒
	Round-trip translation	☐
	Presentation instructions	☒
	Document content	☒
	Dynamic content	☐
	Metadata	☐
	Annotations	☐
	Document parts	☒

Required features:

• Tables
  ○ OOXML: section 17.*
  ○ ODF: section 8.1, 15.*

Requirements:

When translating such a document between the ODF and OOXML standards, the result must meet structure-related requirements in addition to preserving the visual appearance, as shown in the "Simple text formatting" use case. Document parts must be consistently translated to enable users to edit hyperlinks, table cells and even complex nested tables after converting the document's format.

Conclusion:

The translation of table structures between ODF and OOXML is supported in most cases. Problems appear when using table background patterns (not supported by ODF) as well as sub-tables (not supported by OOXML). Another problem is ODF's lack of support for certain layouts, such as the "right to left" layout. Such layout options could be emulated within the options available to ODF, but even so would still require a complex translation. ODF's lack of support for document themes which OOXML uses frequently could cause information loss during translation.

These differences restrict the translatability of tables between the two standards.

4.1.5 Itemization and numeration

Description:

Besides common table functionality, other important features commonly used in office documents are numerations and lists which are often used to present structured information. Employee A sets up a shared online workspace to facilitate the exchange of larger files related to the joint venture. He sends a document explaining the workspace login procedure to his contacts in federal state B. Created in format A, the document describes the required tasks in a few steps. The employees at federal state B open the document using an application supporting format B and see the following:

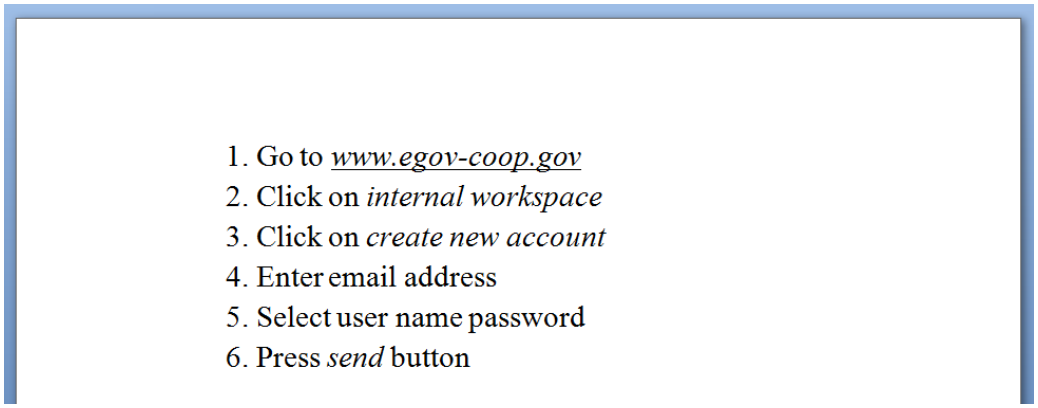


Figure 10: Numbered items

Implementation:

The example shown in Figure 10 contains a simple list of instructions typed in plain text. The instructions are listed using simple numerals and special characters (".") as separators.

Use case name: Itemization and numeration		
Translation type and fidelity		
	One-trip translation	☒
	Round-trip translation	☐
	Presentation instructions	☒
	Document content	☒
	Dynamic content	☒
	Metadata	☐
	Annotations	☐
	Document parts	☐

Required features:
• Itemization and Numeration
  ○ OOXML: section 8.3; 17.9
  ○ ODF: section 4.2, 14.9		

Requirements:

During a translation between both standards it should be possible to retain the index values and structural order in the numeration and list parts of the document.

Conclusion:

Due to the ambiguous wording of the ODF standard for numeration, multiple interpretations of certain itemization and numeration properties are possible. Both formats have multiple ways of applying numbering to text segments. Maintaining visual fidelity would probably call for relatively complicated transformation methods between the two standards, even if the logical hierarchy of different layers of indices was preserved.

The translation of itemization and numeration properties between the standards ODF and OOXML is described in more detail in section 5.2.5.

4.1.6 Index and table of contents

Description:

After the creation of the shared workspace and an initial budget approval, employee A starts to create a document setting goals, explaining results and covering different topics in different chapters. To give a rapid overview of the content and structure and to facilitate navigation, an automatically generated index is added to the document. The document created is saved in format A. Employee B opens the document in his format B-supporting application, removes chapters, and sends the revised version (in format B) to his colleagues.

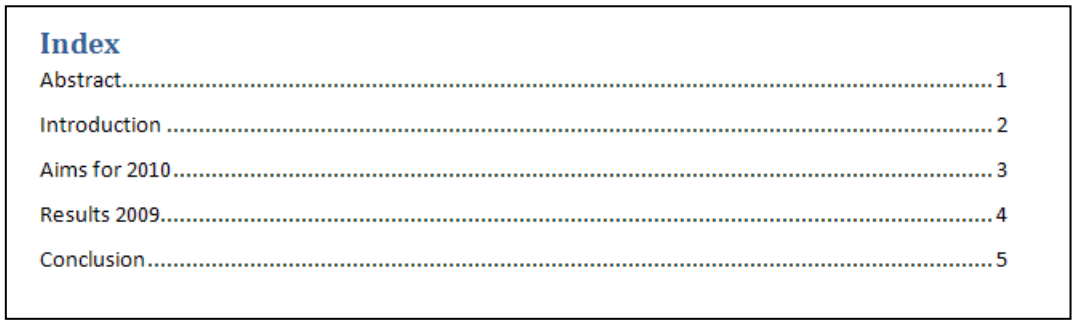


Figure 11: Auto-generated index

Implementation:

In addition to continuous text and structural and presentation features, large documents can contain indices and tables of contents, to enhance readability and to make them more human searchable. Indices like the one shown in Figure 11 should display the document's structure based on headings and page numbers as well as figures and tables based on their captions. Indices should be generated automatically by the available word processing application and kept updated as necessary.

Use case name: Index and table of contents		
Translation type and fidelity		
	One-trip translation	☒
	Round-trip translation	☐
	Presentation instructions	☐
	Document content	☐
	Dynamic content	☒
	Metadata	☐
	Annotations	☐
	Document parts	☐

Required features:
• Indices
  ○ OOXML: section 17.16
  ○ ODF: section 7.*		

Requirements:

The table of contents should be adapted automatically; deleted chapters should no longer appear in the index, and the page numbers of the remaining parts of the document should be updated. The main requirement in this scenario is that the table of contents and index from a format A document can be correctly translated into a corresponding table of contents and index in a format B document.

Conclusion:

Although the two document formats differ in their approaches to the generation of tables of contents and indices, they do indeed offer comparable levels of support for this feature. Implementations must take into account the different models, which makes the translation much more complex, especially when documents combine the available models. A more detailed view of index handling is given in section 5.2.6.

4.1.7 Metadata and settings

Description:

When a document created by employee A using English vocabulary, punctuation and spell checking is saved in format A and sent to employee B, who uses an application supporting format B, his application should immediately recognize which language was used when creating the document.

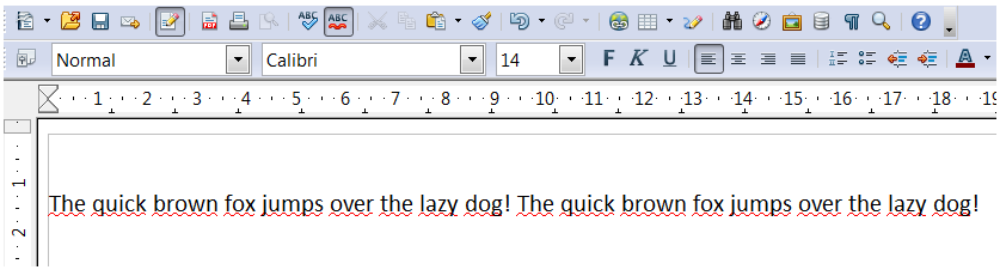


Figure 12: English text with German settings

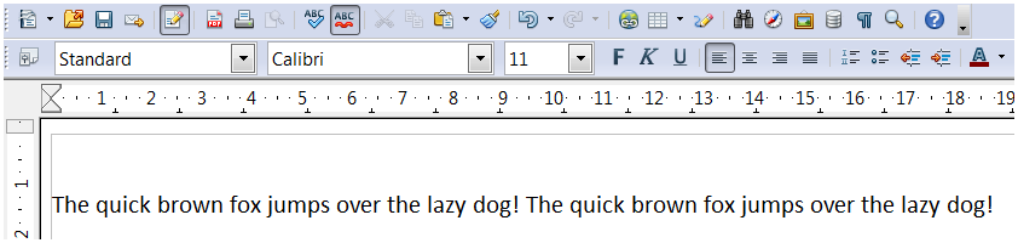


Figure 13: English text with English settings

```
<style:default-style style:family="paragraph">

    <style:text-properties style:use-window-font-color="true" style:font-name="Calibri"
        fo:font-size="11pt" fo:language="en" fo:country="GB"
    />
</style:default-style>


    <w:settings
        xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main">

        <w:themeFontLang w:val="en-GB"/>

    </w:settings>
```

Figure 14: Language metadata info

Implementation:

To ensure the accessibility of word processing documents, certain additional information must be stored as metadata. One example of such metadata is the settings indicating the language used in authoring a document. Grammar and spelling-checkers will need this information when working with the translated document.

The document shown in Figure 12 was written in English with an application normally using German as its default language. Thus, the written words are not recognized by the German spelling checker, as shown by the squiggly red lines displayed under each word. In Figure 13, the language settings have been modified, as evidenced by the absence of the red lines denoting misspelling. Excerpt of the documents' metadata files are given in Figure 14 where the position indicating the default language is underlined in red.

Use case name: Metadata and settings		
Translation type and fidelity		
	One-trip translation	☒
	Round-trip translation	☐
	Presentation instructions	☐
	Document content	☐
	Dynamic content	☐
	Metadata	☒
	Annotations	☐
	Document parts	☐

Required features:
• Metadata
  ○ OOXML: section 17.3
  ○ ODF: section 2.*, 3.1		

Requirements:

A target word processing application must be able to correctly interpret a document's metadata if costly errors are to be avoided. Translation tools should ensure adequate mapping or meaningful default mapping of the metatags when translating between standards.

Conclusion:

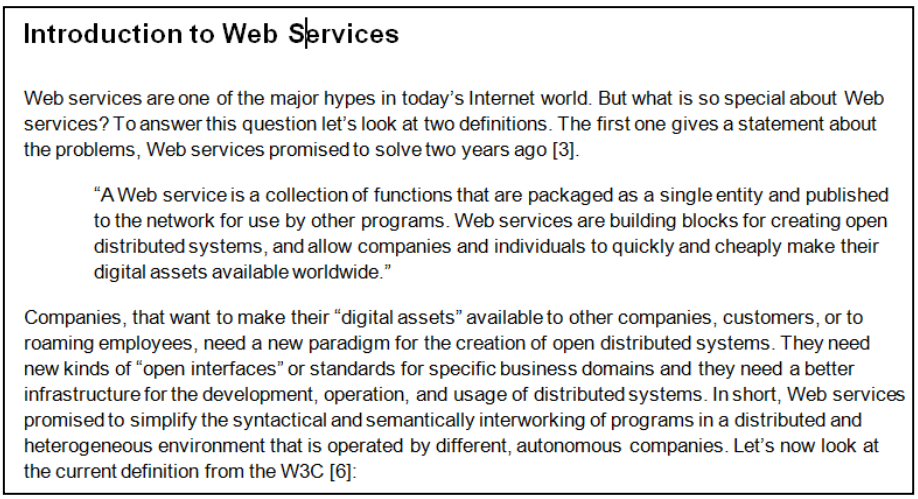
Both standards support different types of metadata. Language information can be adequately translated.

4.1.8 Change tracking and collaboration functions

Description:

The following scenario illustrates how collaboration between different authors using different text processors should proceed.

Employee A, using format A, is planning to publish an article about Web services. Employee B and some other colleagues will all contribute to it, authoring in format B. The first draft of the document will be provided by employee A. The following screenshot illustrates the initial version of the article:



Introduction to Web Services

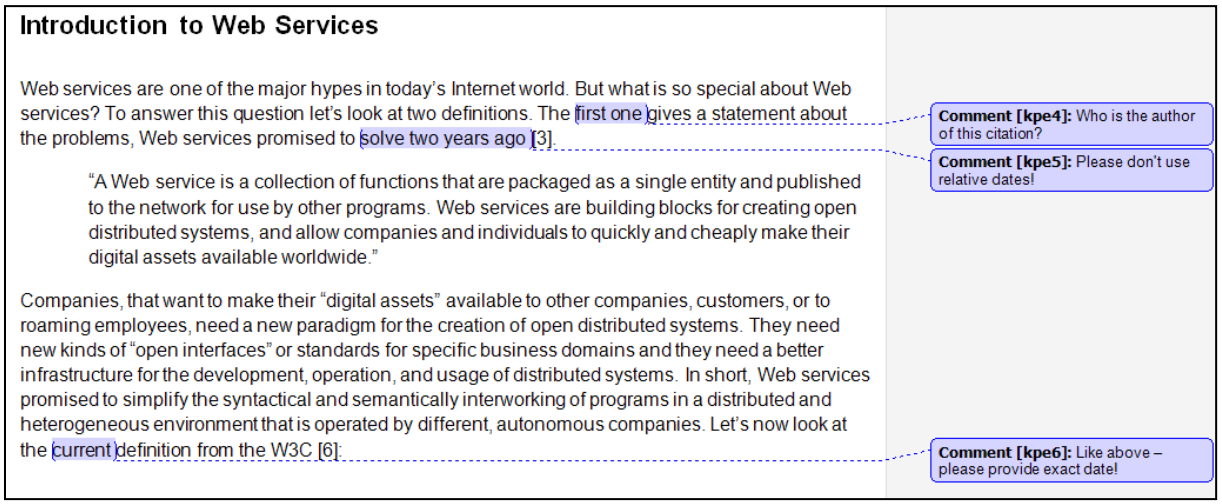
Web services are one of the major hypes in today's Internet world. But what is so special about Web services? To answer this question let's look at two definitions. The first one gives a statement about the problems, Web services promised to solve two years ago [3].

> "A Web service is a collection of functions that are packaged as a single entity and published to the network for use by other programs. Web services are building blocks for creating open distributed systems, and allow companies and individuals to quickly and cheaply make their digital assets available worldwide."

Companies, that want to make their "digital assets" available to other companies, customers, or to roaming employees, need a new paradigm for the creation of open distributed systems. They need new kinds of "open interfaces" or standards for specific business domains and they need a better infrastructure for the development, operation, and usage of distributed systems. In short, Web services promised to simplify the syntactical and semantically interworking of programs in a distributed and heterogeneous environment that is operated by different, autonomous companies. Let's now look at the current definition from the W3C [6]:

Figure 15: Continuous text

Employee A sends this document in format A to his co-author, employee B with a request for comments. The co-author reviews the document using format B's commenting and change tracking features. The comments, shown as colored boxes on the right margin[21], can be applied to paragraphs, words, and even single characters. Comments are marked with the initials of the user entering the comment, with different colors marking comments made by different users. The change tracking function highlights added, edited, moved or deleted text parts and shows the obsolete text parts in colored comment boxes.



Introduction to Web Services

Web services are one of the major hypes in today's Internet world. But what is so special about Web services? To answer this question let's look at two definitions. The first one gives a statement about the problems, Web services promised to solve two years ago [3].

> "A Web service is a collection of functions that are packaged as a single entity and published to the network for use by other programs. Web services are building blocks for creating open distributed systems, and allow companies and individuals to quickly and cheaply make their digital assets available worldwide."

Companies, that want to make their "digital assets" available to other companies, customers, or to roaming employees, need a new paradigm for the creation of open distributed systems. They need new kinds of "open interfaces" or standards for specific business domains and they need a better infrastructure for the development, operation, and usage of distributed systems. In short, Web services promised to simplify the syntactical and semantically interworking of programs in a distributed and heterogeneous environment that is operated by different, autonomous companies. Let's now look at the current definition from the W3C [6]:

Comment [kpe4]: Who is the author of this citation?

Comment [kpe5]: Please don't use relative dates!

Comment [kpe6]: Like above – please provide exact date!

Figure 16: Continuous text with annotations

After employee B has returned the reviewed version of the article, employee A can revise the text by accepting or rejecting the comments and proposed changes.

Implementation:

One of the most important features for editing large documents with multiple authors is called "collaborative functions" which include user-specific comments and tracking of changes. These

[21] The way comments are shown (rendered) depends on the chosen implementation.

functions enable collaborative workflows, allowing document editing and reviewing by multiple participants. The information required for such workflows, including user data, notes or tracked changes, is embedded within the document itself. Proper adoption of such meta-information plays an important role in the collaborative authoring processes.

This type of application, with its workflow support, substantially alleviates the difficulties of revising documents with multiple authors. The foundation for this document lifecycle is the proper conversion of meta-information from one standard to the other, to correctly retain comments and proposed revision information.

Use case name: Change tracking and collaboration functions		
Translation type and fidelity		
	One-trip translation	☐
	Round-trip translation	☒
	Presentation instructions	☐
	Document content	☒
	Dynamic content	☐
	Metadata	☒
	Annotations	☒
	Document parts	☐

Required features:
• Change tracking and document revision
  ○ OOXML: section 17.*
  ○ ODF: section 3.1, 4.6, 8.3, 12.3		

Requirements:

The references to the paragraphs, words and characters made by employee B using format B should be accurately translated into employee A's format A. The information used for change tracking should also reflect the exact editing (such as highlighted changes) in such a way that it can be accurately reproduced, since it is vital that all proposed changes be rendered properly.

Conclusion:

Both document formats offer support for revision-handling, although there are significant differences in the scope of their revision-handling functionality and their approach to the underlying technical details. For example, ODF does not allow for tracking changes made within tables, while OOXML tracks changes to the content of tables as well as changes to the structure of tables themselves. While ODF only records the fact that a text attribute, such as the used text font, has changed, OOXML records the full history of changes made, ensuring the ability to reconstruct the previous text version. Another difference is in the understanding of text comments. While OOXML allows adding comments to arbitrary text ranges, this feature is not supported by ODF. However similar functionality may be provided by inserting notes associated with a point within the text rather than a range. The table “change tracking and document revision” in section 5.2.7 details how collaborative functions could be used when translating between the different document formats.

4.1.9 Forms

Description:

The joint conference being planned uses optimized internal workflow processes. Employee A has designed a digital application form, to avoid the bother of paper-based workflows. The application form saves the data in a structured form, making it easy to extract information such as mailing lists or statistical data. The form is also used by employee B and other employees from federal state B, and thus involves the transfer of forms between computers running different word processing applications. This use case illustrates some simple features commonly used in forms:

Figure 17: OOXML form

Implementation:

Modern word processing documents are tightly integrated into electronic workflows. They serve as static output formats for reports or certificates and, with their extended form functionalities, they can also be integrated as dynamic, data driven front-ends.

The form in Figure 17 contains different textboxes. The form is filled out by an applicant and submitted via a send button which integrates the form’s data directly into applications which further process the data.

Use case name: Forms		
Translation type and fidelity		
	One-trip translation	☒
	Round-trip translation	☐
	Presentation instructions	☐
	Document content	☒
	Dynamic content	☒
	Metadata	☐
	Annotations	☐
	Document parts	☒

Required features:
• Forms
  ○ OOXML: section 17.16
  ○ ODF: section 11		

Requirements:

To pass this form between applications based on ODF and applications based on OOXML, the form's functionality needs to be preserved. Translating the form from one format to the other for processing or viewing should not result in data corruption.

Conclusion:

Translation of forms between ODF and OOXML is likely to prove problematic, since the two technologies diverge strongly in many aspects of form handling. While ODF is directed to the open standard XForms[22] (Version 1.0 from 2004), OOXML uses "form fields" that support insertion of data through "form controls". Although both concepts work with XML structures, the translatability of forms between the two standards is merely low to average.

[22] (W3C, 2003), has been replaced by 3rd edition in October 2007

4.1.10 Vector graphics

Description:

Employee A designs a logo which is embedded in a format A document and sends it to employee B who opens the document using a format B office application. Ideally, the logo should be displayed by the format B application in the same way as it was by the format A application.

Figure 18: Embedded vector graphic

Implementation:

Vector graphics are essential elements of modern document content and presentation, especially for printing purposes. They are flexible in use, editable to a certain extent, and scalable to nearly any size without any need for special expertise in graphics.

Use case name: Vector graphics		
Translation type and fidelity		
	One-trip translation	☒
	Round-trip translation	☐
	Presentation instructions	☒
	Document content	☒
	Dynamic content	☐
	Metadata	☐
	Annotations	☐
	Document parts	☒

Required features:
• Vector graphics
  ○ OOXML: section 8.6
  ○ ODF: section 9.3		

Requirements:

Graphics embedded in documents should maintain their appearance, scaling, and quality when translating documents between the two formats. There should be no discernible difference between graphics presentation under ODF and OOXML. This applies equally to graphics properties such as pixel size, color encoding etc.

Conclusion:

Unlike bitmap graphics which are represented simply through a MIME type and are virtually platform-independent, vector graphics pose bigger translation challenges. OOXML essentially defines its own DrawingML format to which the now obsolete VML (Vector Markup Language) was a precursor. ODF recommends the use of SVG (Scalable Vector Graphics) which is not as rich in features and functionality as DrawingML. The ODF standard merges the SVG namespace with ODF's namespaces, so the SVG objects in ODF documents can't be handled by generic SVG tools and technologies. These types of disparities could pose potential interoperability problems between the two standards in the area of vector graphics.

4.1.11 Generic fields

Description:

To automate recurring tasks, employee A creates a letter template using generic fields that automate tasks such as including addresses for mass mailings, inserting portions of text or the current date. When employee B reopens one of these template-generated letters for review, amendment and eventually for printing, certain fields are auto-completed, which saves him both time and trouble. After reviewing the letter, employee B sends out the invitations, saved in format B.

```
{ DATE  \@ "dddd, MMMM dd, yyyy"  \*
MERGEFORMAT }
```

Figure 19: Field function displaying the current date

Implementation:

The concept of generic fields was introduced to provide text documents with dynamic content. Fields have become one of the most basic tools in preparing document templates. Fields automatically update to include changing data in the document. Combining fields with AutoText creates a powerful documentation "toolbox".

Use case name: Generic fields		
Translation type and fidelity		
	One-trip translation	☒

	Round-trip translation	☐
	Presentation instructions	☒
	Document content	☐
	Dynamic content	☒
	Metadata	☐
	Annotations	☐
	Document parts	☐

Required features:
• Generic fields
  ○ OOXML: section 17.16
  ○ ODF: section 11.3		

Requirements:

The document created by employee A should contain the same generic fields when it is opened by employee B, and function the same way on both ODF and OOXML platforms. The current system date, for example, should be recognized by the application which opens the document and should be displayed correctly in the automatically updated date fields. It is important that applications correctly interpret all formats and conventions.

Conclusion:

Unlike ODF, OOXML allows text fields to contain arbitrary user-generated content. This functionality is used by third-party applications to extend the document's functionality, i.e. by dynamically inserting (meta-)data into a document, or by extracting data in order to perform calculations. While work is underway to add similar functionality to ODF, such functionality thus far cannot be adequately translated.

4.1.12 Font metrics and C-fonts

Description:

After sending out the invitations, a response letter written using the OOXML format and typed in the Cambria font is sent to both employees.

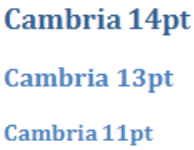


Figure 20: Cambria font

Implementation:

New fonts such as Microsoft's C-fonts (Calibri, Cambria, Candara, etc.) are optimized using ClearType rendering for increased sharpness on liquid-crystal displays. The metrics of the available fonts are used to identify an alternative font in case the primary (C-)font is not available.

Use case name: Font metrics		
Translation type and fidelity		
	One-trip translation	☒
	Round-trip translation	☐
	Presentation instructions	☒
	Document content	☐
	Dynamic content	☐
	Metadata	☐
	Annotations	☐
	Document parts	☐

Required features:
• Font metrics
  ○ OOXML: section 17.8
  ○ ODF: section 2.6		

Requirements:

The document that both employees open in their different applications should line up in all aspects of graphical fidelity. The advantages of ClearType fonts should also be clearly visible when opened in an ODF application and displayed by an appropriate layout engine.

Conclusion:

Due to the use of alternative metrics optimized for ClearType, the appearance of translated documents could differ in terms of line and page breaks. Replacement font types used in alternative document formats such as ODF might require greater or lesser space than the original C-Fonts. One possible solution could be embedding these font types by anchoring them as characters within a text span.

This use case is related to graphic fidelity. Its focus is on features of layout engines and not on translation between document formats.

4.1.13 Equations

Description:

Employee B adds several embedded equations to a document in format B. The equations require special formatting so as to properly represent formulas. Employee B emails the document in format B to employee A who views it using a format A-supporting application.

$$f(x) = \sum \frac{dy}{dx} * y^2$$

Figure 21: Embedded equation

Implementation:

In ODF equations are described using the W3C recommendation MathML[23] and anchored as part of drawing elements within or between text paragraphs. With the additional semantic content definition (in the form of semantic tags and annotations) provided by MathML, equations could also be communicated in different ways. MathML encodes the notational structure of an expression in a sufficiently abstract way to facilitate rendering to various media. Thus, the same presentation markup can be rendered with relative ease on screen in either wide and narrow windows, in ASCII or graphics, in print, or it can be enunciated in a sensible way when spoken.

Equations in OOXML are described in the shared Office MathML Markup Language (OMML) language. These equations are embedded in OOXML documents. They support features such as revision markings, images and regular styles and formatting found in regular WordprocessingML. OMML can be transformed into MathML via XSLT.

Use case name: Equations		
Translation type and fidelity		
	One-trip translation	☒
	Round-trip translation	☐

[23] (W3C, 2003)

	Presentation instructions	☒
	Document content	☒
	Dynamic content	☐
	Metadata	☐
	Annotations	☐
	Document parts	☐

Required features:
• Equations
  ○ OOXML: section 8.6; 17.5
  ○ ODF: section 9.3		

Requirements:

The equations employee B inserts in the document should be displayed on the format-A target system in a form equivalent to that created by employee B. Equation elements may not be omitted, swapped or placed in the wrong position.

Conclusion:

Mathematical content such as equations is represented via MathML in ODF, even though ODF does not import or reference the MathML schema definition. OOXML implements the shared markup language OMML for handling mathematical equations. In OOXML shared parts types can refer to both MathML and OMML. For this reason the translatability between both standards with XSLT is quite high. Change tracking is not possible in MathML.

4.2 Spreadsheets

4.2.1 Listing and structural features

Description:

Employee B uses a spreadsheet using format B to store contact information. The spreadsheet has 5 columns and about 400 entries. The top row contains the column titles: *first name*, *surname*, *address*, *notes* and *date of birth*. To facilitate navigation, the top row is fixed, and will not move while scrolling down the rows. Figure 22 shows an excerpt from the spreadsheet. Employee B emails the spreadsheet to employee A who opens it for editing using format A.

	A	B	C	D	E
1	First name	Surname	Address	Notes	Date of birth
385	Mark	Twain	Trafalgar Square 13, 38163 Hampton beach	**Call urgently!**	14 March 1967
386	Frank	Ross	Mossham Street 19, 27357 Hackleborough	Awaiting action	01 June 1957
387	Sandra	Townsend	unknown	No reply	21 September 1970

Figure 22: Address list

Implementation:

One of the main applications for spreadsheets is the listing and structuring of large amounts of data in sortable tables. Presentation instructions can define the frames, shading and colors used for highlighting and structuring certain parts of the spreadsheet. This use case illustrates the most important functionalities used in spreadsheets. The graphical characteristics of this sheet include its fixed top row, the grey shading of the top row, the colored text in a single cell and the highlighting colored frame on a complete row. The last column uses date formatting which formats any entry as a date.

Use case name: Listing and structural features		
Translation type and fidelity		
	One-trip translation	☒
	Round-trip translation	☐
	Presentation instructions	☒
	Document content	☒
	Dynamic content	☐
	Metadata	☐
	Annotations	☐
	Document parts	☐
Required features: • Formatting ○ OOXML: section 8.4, 12, 17.4, 18.*, 21.2 ○ ODF: section 2.3.4, 6.7, 9.3, 14.7, 15.*		

Requirements:

When employee A opens employee B's list, it should remain obvious that certain elements, such as the row marked red or the red text, are more relevant than others. All the applied presentation characteristics created in format B must be reproduced accurately in employee A's format A application.

Conclusion:

Though certain non-vital features such as shared formulas are not supported by both standards, and features like "cell protection" can only work under certain preconditions in OOXML, more important features such as highlighted cell borders, background images and the assignment of formulas and functions to particular cells are well-translatable. For this use case, the level of translatability with respect to preservation of content and presentation is high.

4.2.2 Formulas and calculation

Description:

Employee B uses a spreadsheet template to place orders for new computer equipment. Employee B uses format B, while the IT department handling the order uses format A.

Product ID	Description	Quantity	Unit Price	Line Total
A1234	Flatscreen	1	275,00	275,00
A1235	Optical Mouse	1	15,00	15,00
A1236	Keyboard Standard	1	17,00	17,00
			SUBTOTAL	307,00
			PST 6,50%	-
			GST 3,20%	-
			SHIPPING & HANDLING	-
			TOTAL	307,00
			PAID	-
			TOTAL DUE	307,00

Figure 23: Spreadsheet based invoice template

Implementation:

In addition to storing and organizing data, spreadsheets are a powerful tool for managing complex and dynamic calculations. Within a spreadsheet, any cell can contain a formula which references the values of other cells using row and column numbers.

Use case name: Formulas and calculation		
Translation type and fidelity		
	One-trip translation	☐
	Round-trip translation	☒
	Presentation instructions	☒
	Document content	☒
	Dynamic content	☒
	Metadata	☐
	Annotations	☐
	Document parts	☐

Required features:
• Calculation
  ○ OOXML: section 18.*
  ○ ODF: section 8.1		

Requirements:

The essential part of this spreadsheet consists of a table for the invoice line items and a self-totaling field for calculating the total cost of the items ordered. Each time a new line item is added, the total due field is updated automatically. Translation of calculation spreadsheets should preserve formula logic as well as presentation and layout information.

Conclusion:

One problem likely to arise when translating spreadsheets is that formula-evaluation is generally application dependent; calculations may work differently if used in different applications. Possible workarounds for such difficulties could be:

- The use of self-written formulas as against the native “out of the box” ones provided by the application which might pose problems during adaptation to non native platforms.

- The mapping of formulas to specific programming/script languages.

The general underlying problem is the lack of a standard for formulas; such a standard would go a long way towards alleviating formula incompatibilities. The OOXML standard includes a documented formula syntax, but ODF does not include a standardized syntax for formulas. The ODF 1.2 specification (currently under development by OASIS) will include a standardized Open Formula syntax, which may enable implementers to more reliably map formulas between ODF and OOXML spreadsheets. This is less of a conversion/mapping problem than an end user inconvenience. Further details are given in section 5.3.3.

4.2.3 Embedded spreadsheet documents

Description:

Employee B wants to pass on information to employee A contained in a format-B spreadsheet. Instead of recreating the portion of the spreadsheet he wants to send, he simply embeds the pertinent spreadsheet information in a text document containing a note and instructions as shown in Figure 24.

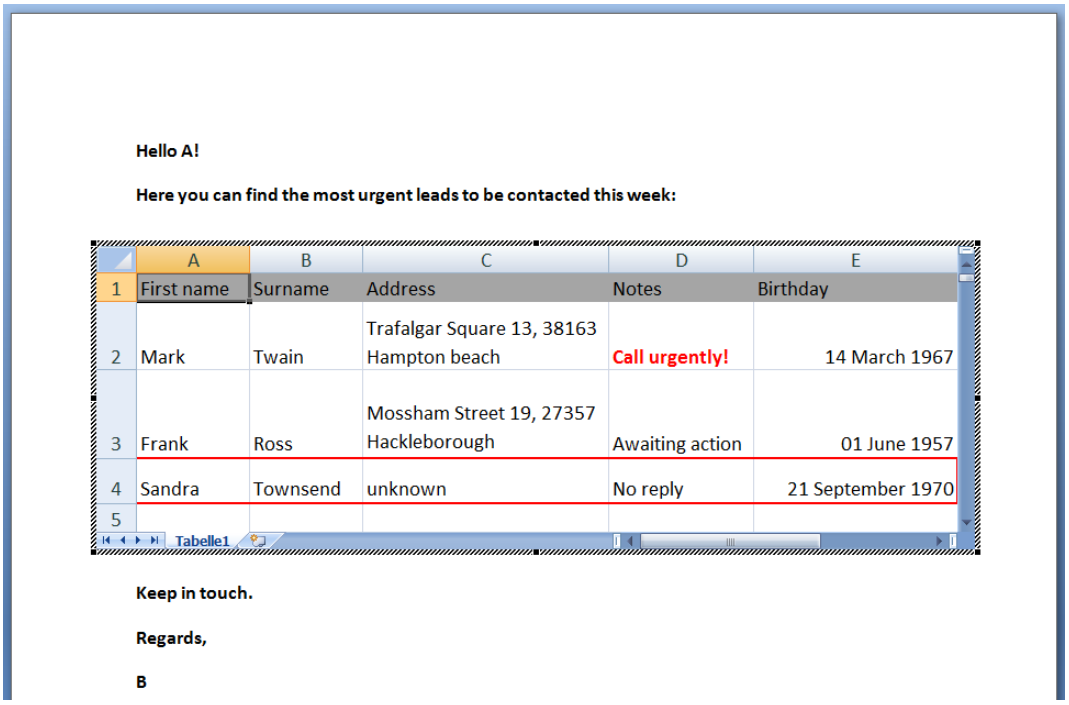


Figure 24: Spreadsheet embedded in a word processing document

Implementation:

An obvious advantage of this approach is that the data in the embedded spreadsheet can be edited and manipulated directly as a dynamic source by the spreadsheet engine rather than being handled statically.

ODF accomplishes this by making use of the <text:insertion> element which contains the information required to identify any insertion of content. Placing a frame within the text area, such as a drawing shape in which a spreadsheet has been embedded, can also be used to create the same effect.

OOXML proposes two options for embedding a spreadsheet within a text document:

- Embedded Packages - Two documents (in this case: a SpreadsheetML document embedded in a WordprocessingML document) are stored together in a format defined by OOXML as an *embedded package*.
- Embedded Objects – The data stored in the object is identified by a unique string (ProgID) which identifies the kind of object data to be embedded.

Use case name: Embedded spreadsheet documents		
Translation type and fidelity		
	One-trip translation	☒
	Round-trip translation	☐
	Presentation instructions	☒
	Document content	☒
	Dynamic content	☐
	Metadata	☐
	Annotations	☐
	Document parts	☒

Required features:
• Embedded spreadsheets
  ○ OOXML: section 15.2
  ○ ODF: section 8; 9.3		

Requirements:

When employee A opens the document containing the embedded spreadsheet, he expects all edited features of the spreadsheet such as color boundaries and highlighted text to be presented exactly as they were when employee B saved the original spreadsheet. For example, the date format needs to be maintained exactly, since an incorrect representation of the original date data could lead to confusion or errors.

Conclusion:

Translatability of embedded objects between ODF and OOXML is not confronted by any major barriers; both standards support Object Linking and Embedding (OLE) as well as alternative image representations of linked objects. Slight translation difficulties may occur in the latter case, since when representing alternative images OOXML may refer to elements of the deprecated VML format which is not an open standard.

4.2.4 Simple text formatting and embedded documents

Description:

Employee B creates a spreadsheet containing several sample newsletter layouts, and saves it in format B before sending it to employee A who opens it with his format A-supporting application.

4

	A	B	C
1	**Description**	**Comments on layout**	**Layout sample**
2	**Newsletter (Version 1)**	**1. Bullet Points** **2. Date on the left upper corner** **3. Signature.** **4. Additional sub-header.**	12 June 2009 Informational Mail Attention! Please take not of the following updates to our event: ✓ Room facilities are finally organized. ✓ The website is now up and running. ✓ Some Papers have been submitted and are in the review process. ✓ Speakers for presentations are still sought. *Signed,* *A, employee of federal state A*
3	**Newsletter (Version 2)**	**1. Continuous text.** **2. Colleagues addressed directly.** **3. Signed with surname.** **4. Space for signature.** **5. Date includes day.**	Internal Mail June 12, 2009 Dear colleagues, The preparations of the event on cooperative eGovernment are proceeding as planned. Room facilities are organized, the website is up and running and some papers have already arrived for publication and the agenda looks sound. Still, we need further help in some areas including the speakers and presentations as well as VIP invitations. Best regards *A*
	Newsletter (Version 3)	**1. Simple bullet points.** **2. Only month shown in date.** **3. Signed with name and position.**	June-08 To whom it may concern: Please take not of the following updates to our event: Room facilities are finally organized. The website is now up and running. Some Papers have been submitted and are in the review process. Speakers for presentations are still sought.

Figure 25: Spreadsheet with simple text and embedded documents

Implementation:

In spreadsheet documents, portions of text are often included as cell content. The use case illustrates one such scenario which is also associated with the formatting and inclusion of graphics.

Spreadsheets often contain formatted text as cell content. This use case illustrates one such scenario which is also associated with formatting and the inclusion of graphics.

The example given in Figure 25 contains three rows and three columns. Column A contains a short text description. Column B contains comments describing the newsletter layout. Column C contains a short text sample formatted using the proposed layout. In addition to paragraph and word formatting, the sample layout in column C also contains embedded graphic elements. Each layout sample fits into the last cell on the row which bears the scaled down proportions of a letter-format page, and is displayed as a page in miniature. The layout samples included in the sheet can either be linked to or embedded within the document.

<table>
<tr><td colspan="3">Use case name: Simple text formatting and embedded documents</td></tr>
<tr><td colspan="3">Translation type and fidelity</td></tr>
<tr><td rowspan="9"></td><td>One-trip translation</td><td>☒</td></tr>
<tr><td>Round-trip translation</td><td>☐</td></tr>
<tr><td>Presentation instructions</td><td>☒</td></tr>
<tr><td>Document content</td><td>☒</td></tr>
<tr><td>Dynamic content</td><td>☐</td></tr>
<tr><td>Metadata</td><td>☐</td></tr>
<tr><td>Annotations</td><td>☐</td></tr>
<tr><td>Document parts</td><td>☒</td></tr>
<tr><td></td><td></td></tr>
<tr><td colspan="3">Required features: Embedded parts

• Formatting
○ OOXML: section 12.3; 15.2
○ ODF: section 9.3</td></tr>
</table>

Requirements:

In translating the information needed to present this spreadsheet using an ODF application, all presentation instructions settings should be preserved. The graphic elements and images should likewise maintain their original graphical appearance.

Conclusion:

The translatability between ODF and OOXML faces no major barriers as both standards support Object Linking and Embedding (OLE) as well as alternative image representations of linked objects. Translating vector graphics could pose slight problems as mentioned in section 4.2.3.

4.3 Presentation

4.3.1 Simple text formatting

The basic features of presentation documents are quite similar to those of text processing documents. The following scenario describes the common features of presentation documents exemplified by a simple keynote presentation. The presentation was created by employee A using format A, and needs to be revised by employee B using format B.

Description:

Employee B opens the presentation for review and checks to make sure the formatting is correct and there are no grammatical errors.

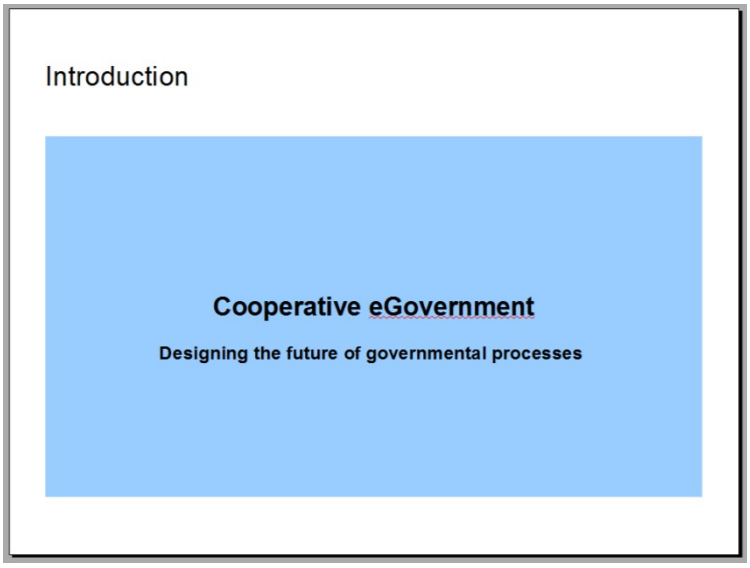


Figure 26: Simple text formatting in presentation documents

Implementation:

The introductory slide makes use of common text formatting features such as centering and bold text.

Use case name: Simple text formatting		
Translation type and fidelity		
	One-trip translation	☐

	Round-trip translation	☒
	Presentation instructions	☒
	Document content	☒
	Dynamic content	☐
	Metadata	☐
	Annotations	☐
	Document parts	☒

Required features:
• Formatting
  ○ OOXML: section 2.3.3, 13.3, 19.*, 21.1
  ○ ODF: section 4.4, 14.6, 15.*		

Requirements:

Any corrections employee B makes in format B, (such as changes to fonts, indentation or layout) should be reproduced without significant discrepancies when employee A reopens the presentation using his format A application.

Conclusion:

The requirements of this use case are relatively easy to translate between the two standards. Details can be found in section 5.4.2.

4.3.2 Itemization and numeration

Description:

Employee A uses his format A application to show the following slide to his head of department before the planned event. Employee B has previously reviewed this slide and sent it back to employee A in format B.

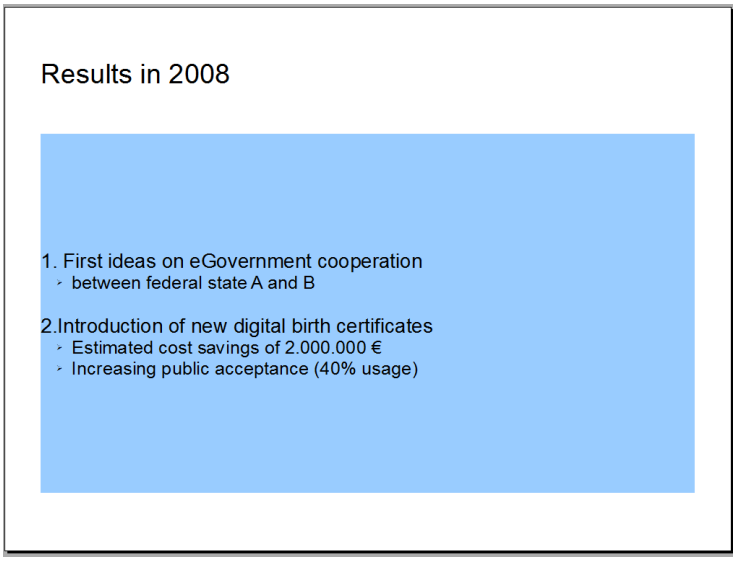


Figure 27: Itemization and numeration in presentation documents

Implementation:

This slide contains a text list similar to that used in word processing applications. The list is comprised of a combination of both numbered bullet point and list items. The bullet points are demarcated by symbols, while the main points are demarcated by numerals.

Use case name: Itemization and numeration		
Translation type and fidelity		
	One-trip translation	☐
	Round-trip translation	☒
	Presentation instructions	☒
	Document content	☒
	Dynamic content	☒
	Metadata	☐
	Annotations	☐
	Document parts	☒
Required features: • Itemization and numeration ○ OOXML: section 13.3.* ○ ODF: section 4.4		

Requirements:

The combination of bullet points and numbered list items should be displayed identically by both applications, since any change in indentation, formatting or symbols used could cause confusion or distortion of facts.

Conclusion:

The minor problems evident in the translatability of word processing documents also apply to presentations because of the way ODF applies the same document root structure to all of its documents. In this use case, however, translatability between the two standards is on a high level.

4.3.3 Positioning and layout

Description:

As in the previous use case, employee A uses his format A application to show the slide below to his head of department before the planned conference. Employee B has previously reviewed this slide and sent it back to employee A in format B.

This slide portrays projected results for two different years. These two years are compared using three short bullet points, and differences between their statistics should be recognizable at first glance.

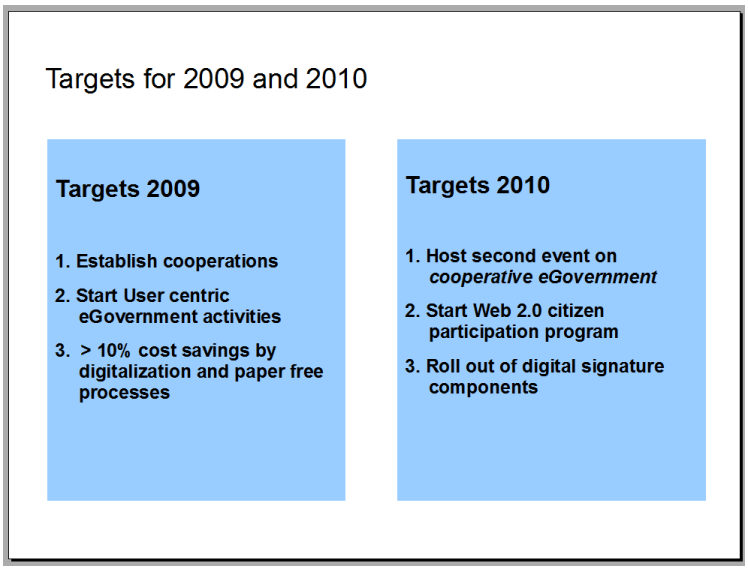


Figure 28: Positioning and layout in presentation documents

Implementation:

The slide contains two sections with graphic elements as backgrounds. Each contains distinctive text. The text in each section is a combination of headers, regular text portions and numbered list items. The two sections differ in content but not, however, in format.

Use case name: Positioning and layout		
Translation type and fidelity		
	One-trip translation	☐
	Round-trip translation	☒
	Presentation instructions	☒
	Document content	☒
	Dynamic content	☐
	Metadata	☐
	Annotations	☐
	Document parts	☒

Required features:
• Positioning and layout
  ○ OOXML: section 13.3.9
  ○ ODF: section 14.15		

Requirements:

In this use case, the presentation is significant. When employee B opens the slide in his format B application, it should display precisely as it did in employee A's format A application. All changes made by employee B should be visible to employee A when he reopens the document and display as they did in the format B application.

Conclusion:

In this use case, translatability is high for both content and presentation. Even so, the translatability of the list featured in this example is limited.

4.3.4 Slide blending and effects

Description:

To enhance the presentation, employee B applies visual animation effects as slide transitions using his format B application. Employee A then reviews the presentation shortly before a board meeting, using his format A application.

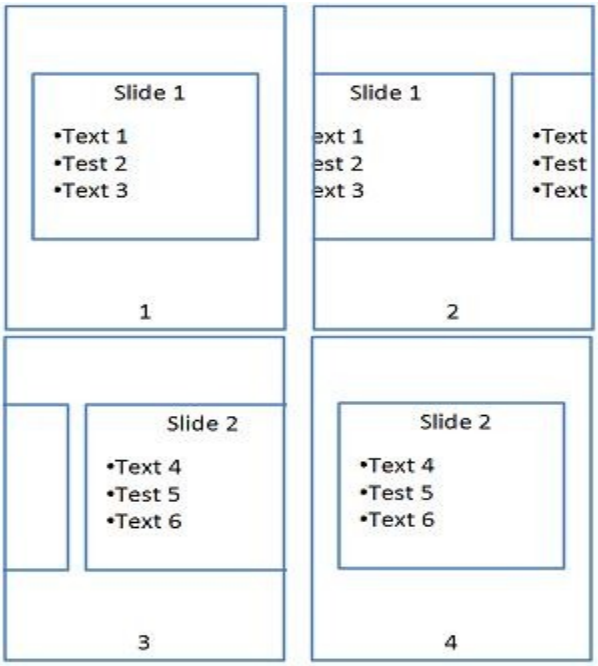


Figure 29: Slide blending and effects in presentation documents

Implementation:

Instead of simple transfers from slide to slide, employee B uses blending effects where one slide blends over into another, as in the fades or "push" transitions illustrated in Figure 29. Animation transitions make the slide changes appear more fluid and give the presentation a smoother overall look.

Use case name: Slide blending and effects		
Translation type and fidelity		
	One-trip translation	☐
	Round-trip translation	☒
	Presentation instructions	☒
	Document content	☐
	Dynamic content	☐
	Metadata	☐
	Annotations	☐
	Document parts	☐

Required features:
 	
• Presentation
  ○ OOXML: section 19.5
  ○ ODF: section 13.1; 15.36

Requirements:

The same visual effects should be visible when employee A opens the presentation using format A. If employee B later alters or adds to the effects already applied using a format B application, such changes should be reflected the next time employee A reopens the document using format A. A roundtrip conversion should be possible.

Conclusion:

Certain features such as time line functionality or transitioning slides along Bezier curves or polylines are not supported by ODF. OOXML provides a far richer lineup of features which are only marginally translatable, or indeed impossible to transform into ODF. This makes for restricted translatability between the two standards with regard to animated slide transition features.

4.3.5 Animations

Description:

For better visualization of the quoted statistics, employee A uses his format A application to add animations to a slide. He then emails the slide to employee B for review.

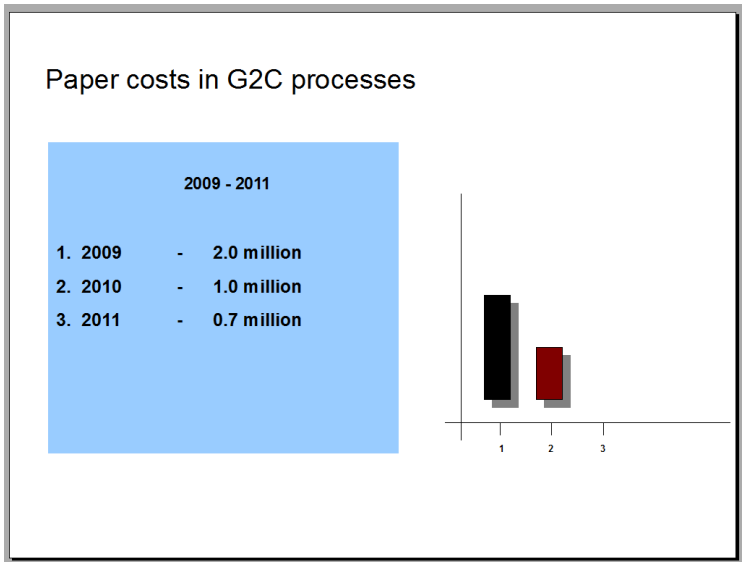


Figure 30: Sample animation in presentation documents

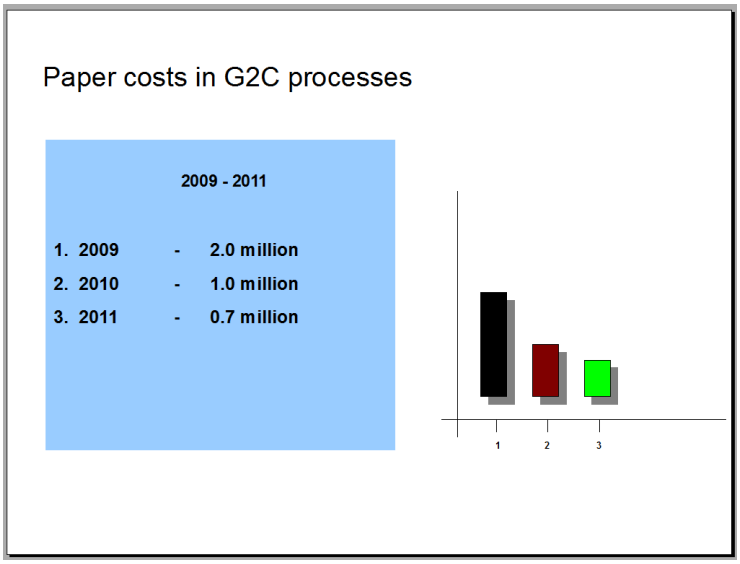


Figure 31: Sample animation in presentation documents

Implementation:

The bars shown in Figure 31 seem animated as they appear one by one with the help of graphic effects which are triggered by a mouse click or shown at timed intervals. The embedded animation is visible for as long as the slide is active.

Use case name: Animations		
Translation type and fidelity		
	One-trip translation	☐
	Round-trip translation	☒
	Presentation instructions	☒
	Document content	☐
	Dynamic content	☐
	Metadata	☐
	Annotations	☐
	Document parts	☐

Required features:
• Presentation
  ○ OOXML: section 19.5
  ○ ODF: section 9.7		

Requirements:

Employee B should be able to replay these animations in a format B environment without noticing any difference; any changes employee B makes to the animations should also be reproducible in employee A's format A environment.

Conclusion:

Both standards have a well-developed set of tools to animate graphic elements. There could be slight difficulties in translatability between applications since animations based on OOXML can be manipulated with finer granularity than those based on ODF. This imposes more constraints on the translation of ODF based applications. One possible way of circumventing some of these setbacks is through the use of SMIL (Synchronized Multimedia Integration Language), which offers a common animation platform for the two standards.

4.3.6 Diagrams

Description:

Employee A creates a slide comparing project results across three years. Employee B has been asked to add the new slide to the keynote presentation.

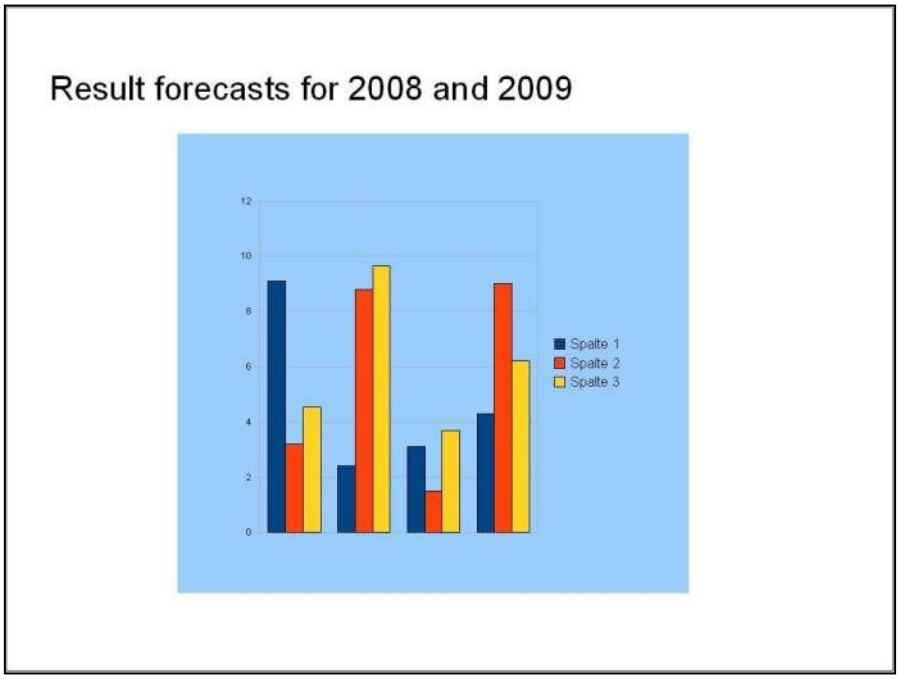


Figure 32: Diagram in presentation documents

Implementation:

Presentation documents can contain simple embedded graphics, called shapes in ODF. Diagrams used in presentation documents in the case of ODF are basically drawing shapes which differ only in their attribute/style-family elements. Presentation shapes are assigned presentation attributes with

a style from the “presentation” family, while drawing shapes are assigned drawing attributes with a style from the “graphic” family. In addition, presentation shapes are further classified based on usage. Examples of such classifications include “text”, “graphic” or, as shown in Figure 32, “chart.”

Use case name: Diagrams		
Translation type and fidelity		
	One-trip translation	☒
	Round-trip translation	☐
	Presentation instructions	☒
	Document content	☒
	Dynamic content	☐
	Metadata	☐
	Annotations	☐
	Document parts	☐

Required features:
• Diagrams
  ○ OOXML: section 14; 12.3
  ○ ODF: section 9.2; 10		

Requirements:

When employee B reviews the document, it is displayed in exactly the way it looks in employee A’s application: The lines, colors and proportions should be the same in both applications.

Conclusion:

The original view would – to a great extent - be retained during a translation between the standards as the translatability between graphic components is high.

4.3.7 Multimedia content

Description:

To make for a more lively presentation, employee B wants to incorporate multimedia content (in this case audio and animated vector graphics) into his slides. Before giving the presentation employee A crosschecks these multimedia slides to ensure everything is working smoothly.

Figure 33: Multimedia content in presentation documents

Implementation:

Employee B has embedded three multimedia elements (audio) each associated with additional graphic elements serving as clickable surfaces (grey rectangles). When an audio is being played, the animated “play” sign appears. When an audio being played is paused, an animated "pause" sign appears. If no audio button is clicked, an animated "stop" sign appears.

Use case name: Multimedia content		
Translation type and fidelity		
	One-trip translation	☐
	Round-trip translation	☒
	Presentation instructions	☒
	Document content	☒
	Dynamic content	☐
	Metadata	☐
	Annotations	☐
	Document parts	☒

Required features:

• Multimedia content and vector graphics
  ○ OOXML: section 15.2.2; 15.2.10
  ○ ODF: section 9.8

Requirements:

When employee A reopens the slide, all media assets should be properly referenced, and the animations should work in the same way they did in the format B application.

Conclusion:

The only means provided by ODF to implement these functionalities is SMIL which is a good alternative to the usual <presentation:animations> element when a mixture of multiple animations are running at the same time. ODF's use of SMIL for certain animation effects is not likely to give rise to any major translatability issues since the schema and syntax of OOXML's PresentationML is loosely based on SMIL.

4.3.8 Master layout

Description:

In order to give the event a "corporate design" employee B creates slide templates to be used by the different presenters. Employee A opens one of the layout templates sent to him by employee B and edits it using a format B application.

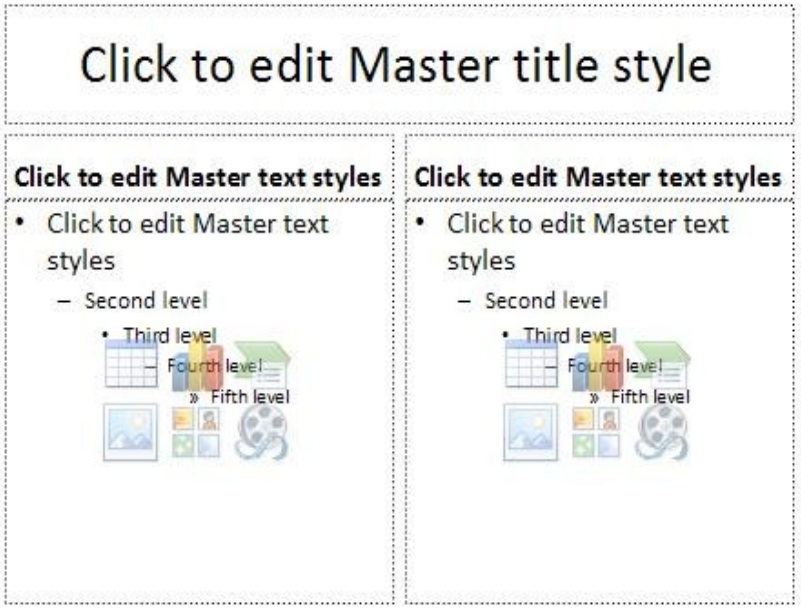


Figure 34: OOXML master slide in presentation documents

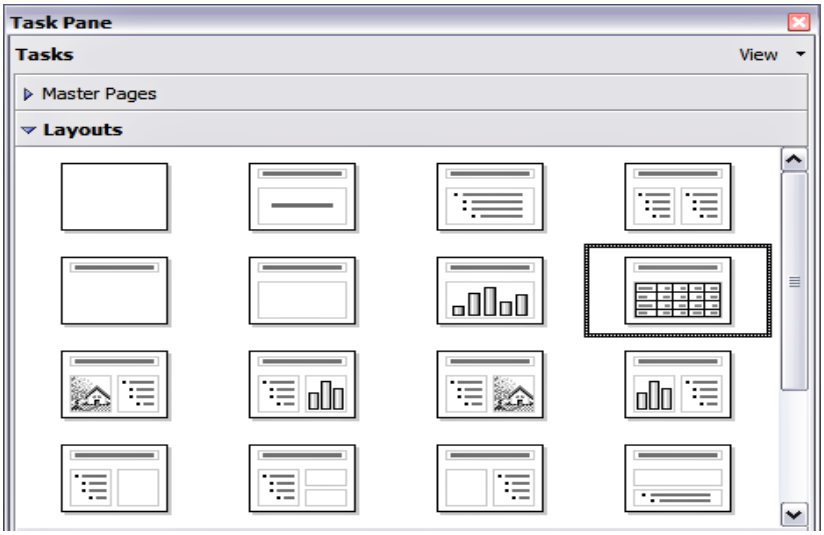


Figure 35: ODF master layout in presentation documents

Implementation:

Employee B uses the master slide to simultaneously edit layout on multiple slides (see Figure 34). Employee A then manipulates the master slide to further adjust the slide layout (see Figure 35).

Use case name: Master layout		
Translation type and fidelity		
	One-trip translation	☐
	Round-trip translation	☒
	Presentation instructions	☐
	Document content	☒
	Dynamic content	☐
	Metadata	☒
	Annotations	☐
	Document parts	☒
Required features: • Presentation Masters ○ OOXML: section 8.5; 13.3; 19.3 ○ ODF: section 9.*; 13.5; 14.*; 15.36		

Requirements:

The changes made by employee A should be reflected in the slide master when employee B reopens the presentation in application B. Employee B should also be able to automatically see the master changes reflected on each individual slide without having to open the master slide settings.

Conclusion:

Translatability between the OOXML master slides and ODF master layouts is very high and satisfies most requirements. For further details, see section 5.4.4.

5 Functionalities and Translatability

5.1 Introduction

This section explains the features needed to implement the use cases described in section 4. The tables in the following subsections summarize the availability of various features for each of the two document formats as well as offering an estimate of the "translatability level" of the various features, which is defined as follows:

- **Low:** Either one of the standards does not support this feature at all, or the way the feature is implemented differs so significantly that feature translation is impossible without information loss.
- **Medium:** Features categorized as having a "Medium" translatability are supported in both formats, although some aspects may differ and workarounds may be required. Features marked as "Medium" may support a single-direction translation, but will result in information loss during round-trip translations. The "Notes" column provides further details on each relevant feature.
- **High:** These features are supported by both standards, and round-trip translation should pose no problems.

The characterization of translatability by the above mentioned metric indicates whether it is possible or in general impossible to translate a feature between the standards. It cannot be assumed that a given tool actually has an implementation for all translations, indicated as "high". On the other hand it cannot be excluded that a given tool has a specific implementation for a translation, indicated as "low". Translation rules will always be tool specific.

It is important to note that the focus of this section is to describe the translatability of various document features between formats and not to engender discussion about the relevance of certain features or to make recommendations for the addition or removal of features from one of the standards. All characterizations are focused on strictly conformant OOXML documents. Transitional conformance as described in ISO/IEC 29500 part 4 is not considered. All statements about ODF refer to ISO/IEC 26300.

5.2 Word Processing Documents

5.2.1 Text formatting

This section contains properties which may be applied to text in word processing documents. Both formats support formatting text at the paragraph level as well as finer granularity. OOXML calls this capability a “*run*,” ODF calls it a “*span*”. The following table summarizes the features which appear in the use cases described in section 4.1.

Table 1: Text formatting

Functionality	Sub-functionality	OOXML	ODF	Translata-bility	Notes
Bold text (font weight)		Yes 17.3.2.1	Yes 14.6.3	Medium	In addition to bold, ODF allows font weight to be specified numerically (100-900).
Text borders		Yes 17.3.2.4	No	Low	ODF only supports borders on whole paragraphs.
Whitespaces		Yes 17.15.1.18 17.18.7 IS29500-3 10.	Yes 1.6	Medium	Because certain OOXML elements (such as the “preserve” attribute defined separately in IS29500 – part 3), are not supported by ODF, translatability of this feature could be problematic.
Capitalization					
	All upper case	Yes 17.3.2.5	Yes 15.4.2	High	
	Small caps	Yes 17.3.2.33	Yes 15.4.1	High	
	All lower case	No	Yes 15.4.2	Low	
Text color					
	RGB	Yes 17.3.2.6	Yes 14.7.8	High	
	Background color	Yes 17.3.2.6	Yes 15.4.37	High	
	Based on theme	Yes 17.15.1.20 17.18.97	No	Medium	ODF has no concept of a “document theme”.
	Blinking text	No	Yes 15.4.36	Low	OOXML supports only blinking backgrounds, but no blinking text.
	Text highlighting	Yes 17.3.2.15	No	Medium	Only a limited range of colors is available for text highlighting.
Complex script support		Yes 17.3.2.7	Yes 15.4.13 15.4.14	Medium	The formats differ in how complex scripts (east-Asian or right-to-left scripts) are supported.
East-Asian text					
	Packing two lines into one	Yes 17.3.2.10	No	Low	

Functionality	Sub-functionality	OOXML	ODF	Translata-bility	Notes
	Brackets around two-lined text	Yes 17.3.2.10 17.18.8	No	Low	
	Vertical text	Yes 17.3.2.10	Yes 15.4.42	Medium	ODF supports rotating text by 0, 90 and 270 deg.; OOXML supports only 0 and 90 deg. rotation.
	Emphasis marks	Yes 17.3.2.12	Yes 15.4.40	Medium	ODF offers more fine-grained support. Marks can be placed above or below the text.
Font selection					
	By font name	Yes 17.8	Yes 15.4.13	High	
	By font family	Yes 17.8.3.9	Yes 15.4.14	High	
	Theme fonts	Yes 17.18.96	No	Low	ODF does not support the concept of document themes.
Font effects					
	Emboss	Yes 17.3.2.13	Yes 15.4.26	High	
	Imprint/ Engrave	Yes 17.3.2.18	Yes 15.4.26	Medium	OOXML has an effect termed “imprint” while ODF offers “engrave”.
	Outline	Yes 17.3.2.23	Yes 15.4.5	High	
	Shadow	Yes 17.3.2.31	Yes 9.5.1	Medium	ODF allows for fine-grained control of text-shadow parameters, OOXML only allows turning the shadow on or off.
Manual specification of run / span width		Yes 17.3.2.14 17.3.2.43	Yes 15.4.41	High	
Italic text		Yes 17.3.2.16	Yes 15.4.25	Medium	ODF supports both italic and oblique text; OOXML makes no such distinction.
Kerning		Yes 17.3.2.19	Yes 15.4.35	High	
Text language		Yes 17.3.2.20	Yes 15.4.23	High	

Functionality	Sub-functionality	OOXML	ODF	Translata-bility	Notes
Enable/ disable spell checking for run/ span		Yes 17.3.2.21 17.15.1.52	No	Low	ODF does not support this feature.
Raised/ lowered text		Yes 17.3.2.24	Yes 15.4.12	Medium	OOXML uses absolute values, ODF uses percentages. This may lead to translation problems.
Strikethrough		Yes 17.3.2.37 17.3.2.9	Yes 15.4.34	Medium	OOXML allows both single and double strikethrough.
Underline		Yes 17.3.2.40	Yes 15.4.28	Medium	OOXML allows both single and double underlining.

5.2.2 Paragraph formatting

In the context of word-processing documents, a paragraph is the smallest unit of text upon which layout is performed. Both document formats support applying the text formatting properties given above on a per-paragraph basis. In fact OOXML simply embeds a run-properties element within the paragraph format whereas ODF paragraph styles may contain both paragraph and text properties.

Table 2: Paragraph formatting

Functionality	Sub-functionality	OOXML	ODF	Translata-bility	Notes
Line height					
	Fixed	Yes 17.3.1.33	Yes 15.5.1	High	
	Minimum	Yes 17.3.1.33	Yes 15.5.2	High	
	Line spacing	No	Yes 15.5.3	Low	
	Font-independent line spacing	No	Yes 15.5.4	Low	
	Automatic	Yes 17.3.1.33	No	Low	OOXML provides a (Boolean) option that specifies "HTML-like" line spacing.
Text alignment (left/ right/ centered/ justified)		Yes 17.3.1.13	Yes 15.5.5	Medium	OOXML supports a range of additional values for Arabic and Thai text.
	For last line in paragraph	No	Yes 15.5.6	No	

Functionality	Sub-functionality	OOXML	ODF	Translata-bility	Notes
	Justify single word	No	Yes 15.5.7	No	
Keep paragraph on same page as following paragraph		Yes 17.3.1.15	Yes 15.5.8	High	
Do not split paragraph across multiple pages		Yes 17.3.1.14	Yes 15.5.10 15.5.9 15.5.8	Medium	OOXML only supports keeping a paragraph on a page without specifying the minimum number of lines and the position of the paragraph.
Tab stops		Yes 17.3.1.37	Yes 7.12.6	High	
	Position	Yes 17.3.1.37	Yes 7.12.6	High	
	Type (left, center, right, decimal)	Yes 17.3.1.37	Yes 7.12.6	Medium	ODF only supports 2 types (left and right). OOXML does not support specifying the decimal character.
	Type (bar, clear, list)	Yes 17.18.84	No	Low	These tab stop styles are supported in OOXML, but their use is discouraged.
	Leader properties	Yes 17.18.72	Yes 7.12.6	Medium	The formats support different kinds of leader styles. ODF reuses the same styles which allows for underline and strikethrough. OOXML supports a fixed list of styles.
	Default tab stop	Yes 17.15.1.25	Yes 15.5.12 14.2	High	
Hyphenation					
	Last word on page	Yes 17.15.1.10	Yes 15.4.44	High	
	Maximum consecutive hyphenated lines	Yes 17.15.1.22	No	Low	
Drop Caps		Yes 17.3.1.11	Yes 15.5.15	Medium	OOXML handles drop caps via specialized text frames. ODF's approach is more straight-forward.

Functionality	Sub-functionality	OOXML	ODF	Translata-bility	Notes
Register truth (same text line distance across multiple pages / columns)		No	Yes 15.2.12	Low	ODF supports a paragraph style attribute which can specify the reference line distance for all paragraphs. This functionality is not supported directly by OOXML. Fixed width tables in OOXML may be able to compensate for this drawback; however there may be difficulties in translatability.
Margins					
	Absolute, relative	No	Yes	Medium	OOXML only supports absolute values for paragraph margins.
	Left/right/ top/bottom	Yes	Yes	Medium	OOXML supports contextual spacing where top/bottom spacing is ignored for identically formatted paragraphs.
First line indent					
	Absolute, relative	Yes 17.3.1.12	Yes 15.5.18	Medium	OOXML only supports absolute values for first-line indentation.
	Based on font size	No	Yes 15.5.19	Low	ODF supports an auto-text-indent property specifying that the first line of a paragraph is indented by a value that is based on the current font size.
Page/ column break					
	Before paragraph	Yes 17.3.1.23	Yes 2.8	Medium	OOXML does not support column breaks as paragraph properties.
Background color		Yes 17.3.1.31	Yes 15.5.23	Medium	OOXML allows using theme color attributes. ODF does not support the concept of a "document theme".
Background pattern		Yes 17.3.1.31	No	Low	
Background image		No	Yes 15.5.24	Low	Background paragraph images are not supported in OOXML.
	Filter	No	Yes 15.5.24	Low	

Functionality	Sub-functionality	OOXML	ODF	Translata-bility	Notes
	Opacity (percent)	No	Yes 15.5.24	Medium	ODF manipulates the opacity of the background image in the form of a percentage, while in OOXML the background color (or filled vector graphics) can be influenced indirectly via alpha color transformations which can be used to modify opacity. Alpha color transformations are expressed as percentages.
Embedded Images		Yes 15.2.14	Yes 9.3.2	Medium	Bitmaps can be easily translated. However, due to discrepancies between SVG (used by ODF) and DrawingML (used by OOXML), there is a high probability that compatibility issues will arise when vector graphics are to be translated.
Borders					
	Top/bottom/ left/ right	Yes 17.3.1.24	Yes 15.5.25	High	
	Between/ bar	Yes 17.3.1.24	No	Low	In OOXML a paragraph may have a "bar" (a border on the "inner" side of the paragraph when a book-like layout is used). Additionally a "between" border can be specified for paragraphs with identical border formatting. ODF allows for merging the borders of consecutive, identically formatted paragraphs.
	Color	Yes 17.3.4	No	Low	OOXML allows for using theme color attributes. ODF does not support the concept of a "document theme".
	Frame effect	Yes 17.3.4	No	Low	
	Shadow effect	Yes 17.3.4	Yes 15.5.28	Medium	ODF offers more fine-grained control of shadow parameters.
	Spacing	Yes 17.3.4	Yes 15.5.27	High	
	Width	Yes 17.3.4	Yes 15.5.26	High	

Functionality	Sub-functionality	OOXML	ODF	Translata-bility	Notes
	Type	Yes 17.18.2	Yes 15.5.26	Medium	OOXML documents can specify "art borders", a concept not supported by ODF. Common styles (single/ double) are supported by both formats.
Padding		Yes 17.3.1.11	Yes 15.5.27	High	
Shadow		Yes 17.3.2.31 17.3.1.29	Yes 15.5.28	High	
Line numbering		No	Yes 14.9.1	Low	OOXML only supports line numbering on a per-section level, not as a paragraph setting. Individual paragraphs can be exempted from line numbering.
	(Re-)set start value	No	Yes 15.5.31	Low	
Vertical alignment (top, middle, bottom, baseline)		Yes 17.3.1.39 17.18.91	Yes 15.27.11		
Asian / complex text layout properties					
	Add space between Asian, ctl and western text	Yes 17.3.1.2	Yes 15.5.32	Medium	OOXML allows for specifying extra spacing between Asian and Roman text as well as Asian Text and Numbers. ODF allows for spacing between Asian, ctl and Western text (but not numbers).
	Allow punctuation to hang into margin	Yes 17.3.1.21	Yes 15.5.33	High	
	Snap to layout grid	Yes 17.3.2.34	Yes 15.2.21 15.5.38	High	
	Line breaking behavior (strict / auto)	Yes 17.3.1.16	Yes 15.5.34	Medium	OOXML allows more specific settings (kinsoku).
Writing mode (lr/rl/tb)		Yes 17.3.1.6	Yes 15.2.19	Medium	OOXML only supports setting paragraph properties to either right-to-left or left-to-right.
Text frames		Yes 17.3.1.11	Yes 9.3	High	

Functionality	Sub-functionality	OOXML	ODF	Translata-bility	Notes
	Suppress overlap	Yes 17.3.1.36	Yes 15.30.5	Medium	In ODF chart text label overlaps may be suppressed. In OOXML this feature is supported with reference to drawing objects. If a text is treated as a drawing object (for example by being grouped with a text) this feature can be used.
Lists		Yes 17.9	Yes 4.3	Medium	Since both formats offer multiple ways of applying numbering information to text segments, an implementation will most likely require fairly complex processing in order to retain the best possible graphical fidelity.

5.2.3 Header and footer

OOXML and ODF both support the definition of header and footer. While OOXML assigns them to the whole document or to single sections, ODF aligns them with the concept of a master page. OOXML supports multiple content types; ODF supports textual headers and footers. Both standards use the terms "header" and "footer" in a slightly different way. To display additional content types than text on the top or bottom of a page, in ODF this content has to be associated with the page instead with the header and footer.

Table 3: Header and footer

Functionality	Sub-functionality	OOXML	ODF	Translata-bility	Notes
Content type		Yes 11.3.6/9	Yes 14.4	Medium	ODF supports text only but other content can be added as part of the master page.
Properties	Separate definitions for right, left, first page	Yes 17.10	Yes 14.4	Medium	ODF allows separate definitions for right and left pages.
Formatting		Yes 17.6.11	Yes 14.3 15.3	Medium	ODF allows formatting headers and footers while OOXML allows formatting pages including headers and footers.

5.2.4 Tables

Both OOXML and ODF support the insertion of tables inside a document. Both formats allow table cells to span across multiple rows and / or columns and provide detailed control over the display of table elements. The table below covers the table features from the use case in section 4.1.4 and highlights further areas where functionality varies between the document formats.

Table 4: Tables

Functionality	Sub-functionality	OOXML	ODF	Translata-bility	Notes
Table properties					
	Right-to-left (rtl) layout	Yes 17.7.6.1	No	Medium	ODF does not support rtl-layout for tables. However the functionality can be emulated by appropriately reversing the cell order.
	Alignment of whole table (left, center, right, auto, indented)	Yes 17.4.29	Yes 15.8.2	Medium	ODF has no support for floating tables. However, this functionality may be emulated by placing a table inside a frame.
	Background color	Yes 17.4.32	Yes 15.8.8	Medium	ODF does not support document themes, so information may be lost in translation.
	Background pattern	Yes 17.4.32	No	Low	
	Background image	Yes 17.2.1	Yes 15.8.8	High	
Data alignment	Horizontal / vertical	Yes 17.3.1.13	Yes 15.11.1	High	OOXML aligns cell data in tables embedded in word-processing documents at paragraph level.
Column settings					
	Adjust column width	Yes 17.4.16	Yes 15.9.1	High	
Row settings					
	Adjust row height	Yes 17.4.81	Yes 15.10.1	High	
Cell settings					
	Span multiple columns	Yes 17.4.17	Yes 8.1.3	High	OOXML does this via the vMerge element.
	Span multiple rows	Yes 17.4.85	Yes 8.1.3	High	
Sub-tables		No	Yes 8.1.3 8.2.6	Low	ODF supports the concept of sub-tables, e.g. tables embedded seamlessly within a table cell. While the same effect may be reproduced by splitting and rejoining cells in the containing table, this would require a translator who could "render" the complete table internally.

Functionality	Sub-functionality	OOXML	ODF	Translata-bility	Notes
Borders					
	Color / width / style	Yes 17.4.67	Yes 8.3.3 15.8.12	High	Both formats allow the same values as for paragraph borders.
Table headings		No	Yes 8.2.2 8.2.4	Low	OOXML has no way of identifying certain table cells as being part of a table header. It does contain a <tblHeader> element; however this specifies that the affected row should be repeated on every page the table spans.

5.2.5 Itemization and numeration

Since ODF and OOXML differ in the way they handle numbering (e.g. of lists or headings), the following two subsections contain a short discussion of each document format's approach. Numbering in this context includes the handling of bulleted (itemized) lists as both document formats handle them the same way as numbered lists.

5.2.5.1 Numbering in ODF

ODF contains two ways of expressing lists: an approach based on the nesting of the individual XML tags used to define the list (structural approach) and an approach where regular paragraphs are marked as belonging to a list (attribute approach). The numbering and list formatting applied to a list item or heading is determined by a list style associated with the list (or numbered paragraph).

The structural approach is reminiscent of the way lists are constructed in XHTML[24] with specialized tags denoting lists and list items and the list level being determined by the nesting of list tags in the XML representation of the document content. The attribute approach simply annotates regular paragraphs with attributes identifying them as items of a specific list style at a certain list level. Both approaches are functionally equivalent, however only the attribute approach can be used to apply numbering information to headings.

Unfortunately, the ODF standard is worded ambiguously and thus allows for different interpretations of the attribute approach described above. The standard does not specify whether the numbering logically resides with the list style, or if there is a global counter for each list level which needs to be restarted manually. For example, the XML code shown in Figure 36 is rendered as shown in Figure 37 when the numbering resides with the list style. However, when a global counter is used, the list would display as shown in Figure 38.

[24] (W3C, 2002)

```
<text:numbered-paragraph text:style-name="L2">
  <text:p>List Item</text:p>
</text:numbered-paragraph>
<text:numbered-paragraph text:style-name="L3">
  <text:p>List Item</text:p>
</text:numbered-paragraph>
<text:numbered-paragraph text:style-name="L2">
  <text:p>List Item</text:p>
</text:numbered-paragraph>
```

Figure 36: Numeration in ODF - XML

1.List Item
A)List Item
2.List Item

Figure 37: Numeration in ODF - Counter associated with list style

1.List Item
B)List Item
3.List Item

Figure 38: Numeration in ODF - Global counter

5.2.5.2 Numbering in OOXML

OOXML has no distinct concept of lists. Instead, it uses an approach similar to the ODF "attribute approach" explained above. List items (and headings) are simply regular paragraphs to which special properties are attached which contain information about list structure (an identifier for the list the paragraph belongs to and its list level) and a reference to the formatting information for the list. Headings are treated in the same way, except that they contain additional information about the heading's outline level within the document.

A detailed explanation of the concepts used for numbering information in OOXML is contained in Part 1, section 17.9 of the OOXML standard. Numbering information may be applied to a paragraph in three different ways.

- In the simplest case, the paragraph is annotated with a reference to a *numbering definition* which in turn inherits the actual numbering settings from an abstract numbering definition.
- Alternatively, a numbering style may be applied to the paragraph via one of two distinct yet equivalent approaches. In both cases, the numbering style is not referenced directly; rather, a numbering definition which references the style via its associated abstract numbering definition is applied as shown above.
- The numbering style may also reference a separate numbering definition.

5.2.5.3 Comparison of numbering and enumeration

Both document formats offer a comparable level of support for numbered and/or bulleted lists. OOXML allows for more flexibility when specifying the formatting of nested numbering. To give an

example: using individual suffixes, prefixes, and separators on each level, in OOXML the third-level heading - 1.2.3 *heading* - looks like:

Section I,2.b) *heading*

ODF allows the specification of one common prefix, suffix, and separator for the whole numbering. Thus using the prefix: "Section ", and the suffix: ")" the example will look like:

Section I.2.b) *heading*

Since both formats offer multiple ways of applying numbering information to text segments, a translation implementation will most likely require fairly complex processing in order to retain the best possible fidelity.

5.2.5.4 Metadata language entries

Under both standards, the code is defined by a two or three letter language code taken from the ISO 639 standard optionally followed by a hyphen (-) and a two-letter country code taken from the ISO 3166 standard.

This is how the default language for a *run* would be specified using OOXML:

<w:lang w:val="fr-CA"/>

This language definition is quite similar for ODF. Metadata for language information can generally be adequately translated from one format to the other.

5.2.6 Indices

Office documents may contain various types of indices, including the table of contents, but also indices of figures, tables, etc. Since the two document formats follow different approaches in the way indices are represented, this section offers an overview of both approaches in subsections 5.2.6.1 and 5.2.6.2.

5.2.6.1 Indices in ODF

ODF supports three different types of indices: tables of content, alphabetical indices and user-defined indices. Each index in turn is composed of two parts: an *index template* specifying all the information needed to generate the index and an *index body* containing a rendition of the index, using standard text processing markup.

The information contained within the index template varies according to the index type. The index template specifies the source material for the index, along with an optional title and a template specifying how the title and each index entry should be rendered.

For example, the table of contents described in the use case 4.1.6 is built from the document's headings. Since this table of contents has no title, the template would not specify one. Each entry is built from:

- The entry's title (the section heading in the document)

- A tab stop
- The page number where the heading appears in the document

ODF has three ways to specify the source material for the table of contents:

- Text outline: the document structure, i.e. the headings and their associated outline level are used to generate the table of contents.
- Index marks: this approach only indexes paragraphs and headings which are explicitly marked with an index mark.
- Styles: the index is built from paragraphs to which certain text formatting styles are applied.

5.2.6.2 Indices in OOXML

In OOXML, the concepts of tables of contents and indices are implemented as dynamic content fields. Thus, a table of content is represented by a TOC field, and its presentation and source material are specified by the field's switches.

The source material may be based on the following:

- Paragraph-outline level. This approach corresponds to ODF's approach to using the document structure.
- Index marks (implemented via TOC fields in OOXML) or bookmarks.
- Styles: This approach is similar to the styles-approach offered by ODF.
- A sequence (commonly used for lists of figures/tables/etc.)

5.2.6.3 Summary

Although the two document formats differ in their approaches to the generation of tables of contents and indices, they do offer comparable levels of support for these features. Implementations will have to take into account the different models, which causes some complexity, especially when documents combine many of the approaches outlined above.

5.2.7 Change tracking and annotations

Both document formats offer support for change tracking and textual annotations in word processing documents. In addition to the common operations, OOXML allows highlighting text regions with a limited set of colors (for more information, see section 5.2.1). ODF's change tracking support is more coarsely-grained than that of OOXML in that formatting changes, including those in tables, are recorded but no information about the previous state is kept so that the previous state cannot be reconstructed by simply rejecting the changes.

Table 5: Annotations

Functionality	Sub-functionality	OOXML	ODF	Translata-bility	Notes
Text insertion		Yes 17.13.5	Yes 4.6.3	Medium	Change tracking in lists may cause problems in ODF.
Text deletion		Yes 17.13.5	Yes 4.6.4	Medium	Changing tracking in lists may cause problems in ODF.
Formatting changes		Yes 17.13.5	Yes 4.6.5	Medium	ODF only records the fact that a change has occurred. No further information is recorded, so it is impossible to fully reconstruct the previous state.
Comments		Yes 17.13.4	No	Medium	OOXML allows adding comments to arbitrary text ranges. This is not supported by ODF, however similar functionality may be provided by inserting notes (associated with a point in the text, not a range).
Text highlighting		Yes 17.3.2.15	No	High	Although ODF does not support text highlighting, the functionality may be emulated by setting the text background color (see the section on text formatting).
Metadata					
	Name	Yes 17.13	Yes 3.1.6	High	
	Date / Time	Yes 17.13	Yes 3.1.9	High	
	Author shorthand for comments	Yes 17.13 17.13.4	Yes 12.3 8.3.3	High	

5.3 Spreadsheets

5.3.1 Introduction

This section describes the properties which may be applied to the elements of spreadsheet documents. For the purposes of this paper, the properties to be examined have been narrowed down to formatting and calculation functions and those in any way related to such.

ODF spreadsheets have tables as root elements. Tables in turn contain rows. Rows are divided into cells by columns. ODF does not differentiate between tables embedded in word processing documents and those which make up spreadsheets. Essentially the same XML structures, nodes and

attributes are used in both cases. The only difference is the <spreadsheet> element used within the <body> element as against the <text> element used in word processing documents.

In a similar vein, OOXML has *workbooks* as root elements. Workbooks contain *worksheets*. These sheets are further divided into a grid of *cells*.

5.3.2 Formatting

The cell is the most elementary unit of a spreadsheet to which properties can be applied. Properties of rows, columns and tables (ODF) or worksheets (OOXML) can also be manipulated

The following table summarizes the features pertaining to spreadsheet formatting for the use cases covered.

Table 6: Spreadsheet formatting

Functionality	Sub-functionality	OOXML	ODF	Translata-bility	Notes
Row fixing		Yes 18.3.1.66	Yes ---	Medium	This functionality can be applied in ODF only by manipulating the horizontal / vertical “split mode” and “split position” attributes via the "seetings.xml" file. This file is not defined within the ODF specification and is application-specific.
Cell / Row background Shading		Yes 17.4.33	Yes 15.11.6 15.10.3	High	
Colored text in a single cell.		Yes 18.3.1.53 18.4.7	Yes 14.7.7 15.4.3	High	
Highlighted color frame on single row		Yes 18.8.5	Yes 15.5.25	High	
Date formatting		Yes 18.17.4	Yes 6.7.7	High	
Graphic cell content					
	Linked	Yes 21.2.2.63	Yes 9.3.2	High	
	Embedded	Yes 21.2.2.63	Yes 9.3.2	Medium	When using embedded images, the use of vector graphics could prove problematic due to the different vector graphic formats used by ODF and OOXML.

Functionality	Sub-functionality	OOXML	ODF	Translata-bility	Notes
Spreadsheet-embedding in other applications		Yes 18.3.1.60	Yes 9.3.7	Medium	A few problems could arise due to OOXML's use of VML - which is not supported by ODF - in certain areas.

5.3.3 Calculation

OOXML and ODF calculations are performed using equations also known as formulas.

In OOXML named formulas are known as functions. Formulas are represented by the text of the formula and the text version of the last computed value for that formula. The return value of a function is specified within the "t"-attribute of the cell containing the formula.

The ODF spreadsheet content model contains a spreadsheet calculation setting for formulas. The presentation of the value of a variable is set using a <text:variable-set> variable setter element in which the attribute <text:formula> contains the formula used to compute the value of the variable field. Settings pertaining to the calculation of formulas are set via the <table:calculation-settings> element. The <table:formula> attribute generally contains the formula for a table cell.

This section describes the translation of functionality provided by the properties used in applying formulas to cells as well as their behavior and underlying logic operations as used in the use case example given in section 4.2.2.

Table 7: Spreadsheet calculation

Functionality	Sub-functionality	OOXML	ODF	Translata-bility	Notes
Assigning formulas/ functions to a cell		Yes 18.3.1.40	Yes 8.1.3	High	
Manual/ automatic calc. mode		Yes 18.18.4	No	Low	In OOXML the formulas can be executed whenever a cell value changes or when a user initiates an action.
Shared formulas		Yes 18.3.1.40	No	Low	In OOXML primary/ shared formulas are used to cut down redundancy where a formula is used more than once. This functionality is not present in ODF.
Externally referenced formulas		Yes 18.14 18.14.1 18.18.11	No	Medium	In ODF cells can be referenced but not formulas. OOXML allows the direct referencing of both.

Functionality	Sub-functionality	OOXML	ODF	Translata-bility	Notes
Caching of externally referenced workbook		Yes 18.10.1.95 18.14.7	No	Low	External workbooks cannot be referenced in ODF.
Defined names in place of cell references in formulas		Yes 18.17.2.5	No	Low	Names to be used in place of references or formulas do not exist in ODF.
Auto filtering		Yes 18.3.1.2	No	Low	In ODF no tags exist to specify the criteria for which cells in a table should be displayed. Instead cell validation content rules can be defined that determine which content may be allowed in cells.

5.3.4 Additional properties

This table contains an extended list relating to the analysis of the translatability of selected functionalities for spreadsheet documents.

Table 8: Additional spreadsheet functionality

Functionality	Sub-functionality	OOXML	ODF	Translata-bility	Notes
Width adjustment		Yes 18.3.1.13	Yes 8.1.1 15.7.4	Medium	In ODF columns must have fixed width; relative width is only an option, specified as a percentage.
Alignment		Yes 18.8.1	Yes 15.11.1 8.1.3	Medium	In ODF L, R, C, margins exist. Additionally, OOXML offers header and footer margins.
	Vertical alignment	Yes 18.8.1 18.18.88	Yes 15.11.1 8.1.3	High	
	Horizontal alignment	Yes 18.8.1 18.18.40	No	Low	
	Rotation angle/ align	Yes 18.8.1	Yes 15.11.12 15.11.13	High	
Page number		Yes 13.3.3	Yes 6.2.3 15.2.2	High	

Functionality	Sub-functionality	OOXML	ODF	Translata-bility	Notes
Table or worksheet background/ image		Yes 18.3.1.67	Yes 15.5.24	High	
Shadow		Yes 18.8.36	Yes 15.2.9	Medium	OOXML (SpreadsheetML) applications are not required to render according to the "shadow" flag.
Cell border		Yes 18.8.4	Yes 15.11.7 8.1.3	High	
Cell protect		Yes 18.8.33	Yes 15.11.14 8.1.3	Medium	In OOXML cell protection does not take effect unless the sheet has been protected.

5.4 Presentations

5.4.1 Introduction

ODF and OOXML use different approaches to define presentation documents. In ODF, presentation documents are composed of a set of <draw:page> elements within an <office:presentation> element. A <draw:page> element acts as a container for content.

OOXML presentation documents are based on PresentationML (a framework loosely based on SMIL) in which all definitions are stored as a schema (XSD) which can be one of either structural or presentation level data types.

5.4.2 Slides

5.4.2.1 OOXML slides

In OOXML, the transition from one slide to another is performed via animation effects that are displayed in between slides. Slides, layouts and notes can be defined via "masters". These master layout components can be overridden individually by specifying local attribute values within each presentation slide.

Hierarchy and inheritance are central to the concept of slides in OOXML.

5.4.2.2 ODF slides

ODF animation effects are carried out on "presentation shapes" (these are differentiated from drawing shapes by the <presentation:class> attribute).

It is possible to specify multiple effects for each shape within a page. However this could be hampered by the application on which the presentation is running which can in some cases restrict the extent to which this feature can be utilized.

Several effects can also be initiated at the same time via animation groups:

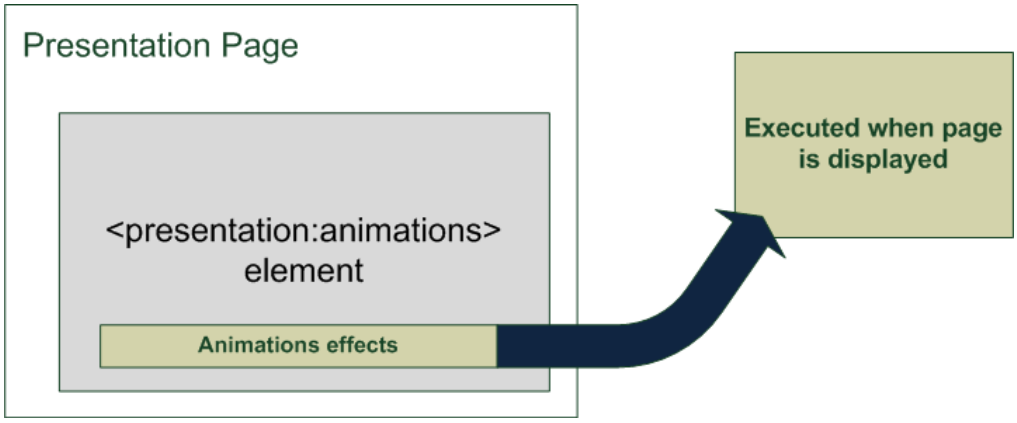


Figure 39 - Animation effects

As an alternative, the animations in ODF presentation documents can be manipulated using the XML based SMIL meta language on which the OOXML PresentationML schema is loosely based.

5.4.3 Text formatting

This section describes properties which may be applied to text in presentation documents, based on the use cases discussed in section 4.3. Text formatting in presentation documents is similar to text formatting in word processing and spreadsheet documents.

Table 9: Text formatting

Functionality	Sub-functionality	OOXML	ODF	Translata-bility	Notes
Bold type		Yes 19.2.1.1	Yes 14.6.3 15.4.32	Medium	In addition to bold, ODF allows font weight to be specified numerically (100-900).
Listing and itemization		Yes 21.1.2.4.1 (19.3.1.5, 19.3.1.35 19.3.1.52)	Yes 7.1	Medium	Since both formats offer multiple ways of applying numbering information to text segments, an implementation would most likely require fairly complex processing in order to retain the best possible graphical fidelity.
Text animation		Yes 19.5 M.3.4.7	Yes 15.15	Medium	ODF sets attributes via <draw : frames> controlling style or SMIL. OOXML applies build animations.
Text language		Yes 21.1.2.3.9	Yes 15.4.23	High	

5.4.4 Master layout

ODF makes use of master pages for creating slides. A “master page” is actually a reference to a specific page layout which is used as a base template when beginning to develop a presentation. This template specifies properties common to each page, such as size, content, headers, and footers, which are displayed on every page in a presentation. ODF specifies that all documents must contain at least one master page element.

OOXML follows a similar principle. In Microsoft Office 2007, these layout templates are known as "slide masters". Slide layouts can override definitions preset by slide masters, and can also be applied to individual Office presentation slides. This approach offers greater layout flexibility than that offered by ODF.

The following table compares presentation document functionality, based on the use cases discussed in section 4.3.8:

Table 10 – Master layout

Functionality	Sub-functionality	OOXML	ODF	Translata-bility	Notes
Layout and positioning		Yes 19.7.15	Yes 14.15	High	
Animations		Yes 19.5.1	Yes 9.7	Medium	OOXML specified animations can be applied in a greater number of ways than ODF specified ones. This allows for finer granularity in creating slide animations.
	Specialized path descriptions	Yes 19.5.4	No	Low	OOXML provides for animation via motion descriptions over polyline or Bezier paths. ODF does not support this.
	Timeline functionality (using time nodes)	Yes 19.3.1.48 19.5.87	No	Low	In addition to inheritance from, or overriding of, master-layouts, OOXML makes use of "timelines" to orchestrate its animations. ODF does not support the concept of timelines.
Slide synchroni-zation		Yes 19.6	No	Low	An update function used by OOXML for synchronizing slides being loaded from SharePoint servers. ODF documents can at most load texts stored in a SQL database if an appropriate driver has been installed.
Applying sounds to slides		Yes 19.5.69	Yes 9.7.1	High	

Functionality	Sub-functionality	OOXML	ODF	Translata-bility	Notes
Diagrams		Yes 20.1.2.2.12	Yes 9.7.2	High	
Slide blending and effects		Yes 19.3.1.50	Yes 9.7 9.8.1 15.36.2	Medium	In ODF specification of multiple effects could become problematic since the application on which the presentation is being run can in some cases restrict the extent to which this feature can be utilized. The restriction varies from application to application.
Multimedia content		Yes 19.3.1.33	Yes 9.8 13. 15.36.10	Medium	In OOXML media can be orchestrated to play in sync with a slides timeline. If the media supplying the sound for instance is a CD other attributes such as track indexes or the start or end track can be specified.
Vector graphics		Yes 20.1 M.5	Yes 9.2.6 14.14.2	Low	Due to the use of different graphic engines, the vector graphics are not translatable. However both ODF and OOXML individually support the representation of vector graphics.
Master layout		Yes 19.2.1.36	Yes 14.4	High	

5.5 Common Aspects

This section covers functionalities spanning multiple document types.

5.5.1 Alternative presentations

Metadata, such as alternative text representations for non-text entities within a document, play an important role not only in granting people with disabilities better access to document content, but also in improving the automated extraction and processing of information contained within a document.

The following table gives a brief comparison of alternative presentations supported by ODF and OOXML.

Table 11: Alternative presentations

Functionality	Sub-functionality	OOXML	ODF	Translata-bility	Notes
Alternative text					
	Images	Yes 18.3.1.56 17.3.3.19	Yes 9.3.9	High	
	Image maps	No	Yes 9.3.11	Low	OOXML does not support image maps.
	Lines / Arrows	Yes 18.3.1.56 17.3.3.19	Yes 9.3.9	High	
	Auto shapes	Yes 18.3.1.56 17.3.3.19	Yes 9.3	High	
	Grouped objects	Yes 18.3.1.56 17.3.3.19	Yes 9.3.9	High	
	Sounds	Yes 18.3.1.56 17.3.3.19	No	Low	Possible alternative: ODF supports alternative text for OLE-Objects which could be video or audio objects.
	Videos	Yes 18.3.1.56 17.3.3.19	No	Low	Possible alternative: ODF supports alternative text for OLE-Objects which could be video or audio objects.
	Charts	Yes 18.3.1.56 17.3.3.19	Yes 9.3	High	
	Text-box, titles, captions	Yes 18.3.1.56 17.3.3.19	Yes 9.3.9	High	
	Links	Yes 18.3.1.56 17.3.3.19	Yes 9.3.10	High	

5.5.2 Custom XML parts

Custom parts of documents contain arbitrary XML markup not necessarily defined by the document's standard itself. OOXML (section 22.5) allows arbitrary XML instance to be stored in a document, and the nodes of that XML instance may be bound to form controls (content controls). ODF does not support arbitrary custom XML parts, so these would be lost in a round-trip to ODF.

6 Conclusion

This White Paper in its two main parts describes typical effects which occur during the mixed usage of documents based on the standardized formats ISO/IEC 29500:2008 (OOXML) and ISO/IEC 26300:2006 (ODF). In the first main part use cases describe the interchange of documents between different office applications supporting different document standards. The use cases focus on situations in which documents are exchanged between different public administrations, and public administrations and citizens respectively. The effects occurring during the interchange, their origins as well as useful ways of avoiding incompatibilities are described. The scope of the research covers word-processing, spreadsheet and presentation documents.

In the second part of the White Paper the two ISO Standards ISO/IEC 29500:2008 (OOXML) and ISO/IEC 26300:2006 (ODF) are analyzed in more detail in terms of the identified functionalities. Each functionality's underlying principles and concepts are analyzed in both standards, collected in a spreadsheet table and compared with regard to their mutual translatability. A three value metric is used to characterize missing, possible and good translatability. References to the corresponding passages in the standards are added to the tables in order to nurture an in-depth understanding of the given characterizations for each specific functionality.

It may be concluded that many of the functionalities, especially those found in simpler documents, can be translated between the standards, while the translation of other functionalities can prove complex or even impossible. Frequently in individual cases it has to be decided, if the conversion of a document is completely translatable, translatable only to a limited extend or not at all. The individual cases are determined by different constraints. First and foremost translatability depends on the document itself together with its characteristics. In addition the application or tool used for the transformation has also to be considered. In this study statements on translatability and its quality in principle are made. As the rules used for transformation are not standardized however, each application is allowed to use its own specific rules. Under certain circumstances specific rules can neglect certain properties and make specific assumptions which could enhance translatability. In addition, the direction of the transformation has to be considered. In many cases a document can be translated without any major loss from one format to the other. Even so, on a round-trip conversion it cannot be guaranteed that the initial document and the document resulting from the conversion will be identical.

For each use case the following issues must have to be considered:

- Why translate a document from one standard to another?
- Which is the optimal document format to be used in the translation?
- Is it necessary to have a round-trip conversion of the document and if so, why?
- Which are the best tools to achieve these goals and who should use them?

With regard to the findings of this White Paper, it should be borne in mind that the two standards were analyzed in the form published by ISO. Between these two versions of the standard there will most likely never be a need for translation. Thus far there is no application completely supporting ISO/IEC 29500-1/2/3:2008. Even the upcoming version of Microsoft Office 2010 will be implemented according to part four (transitional conformance) of the standard to achieve backward compatibility

with prior Office versions. Applications supporting ISO/IEC 26300:2006 have been switching (2009) to the latest OASIS versions of the standard ODF 1.1 and 1.2 (committee draft). These versions, however, are not ISO standards. The White Paper thus gives information on the basic concepts, similarities and differences of the two ISO standards. It demonstrates the possibilities and limitations of the translation of important functionalities between the two document formats. Newer versions and error corrections of the standards have to be considered in the each individual case.

There are other important limitations when translating between both standards. In fact, two versions of the same document saved in separate standards should also look the same and have the same layout. However, from a technical point of view layout will be determined by the rendering engine and the available hardware and software (resolution, text fonts, colors, etc.). Layout thus only indirectly depends on the standards. There could be differences in terms of the inner structure, even when two documents look alike. On the other hand, two documents with different layouts could have the same inner structure and contents. For this reason it is extremely important to be aware of the invariable aspects of those documents that must be transferred between different organizations, and to decide on a document format by taking into account the reasons for document exchange.

Application interoperability is not only determined by the interoperability of the implemented standards. Ambiguous specifications, which partly occur within the standards, reduce the unambiguousness of document implementations as well as the standard conformity of the documents. Also the usage of standard extensions makes translation of documents difficult, even if the extensions are inserted in a standards compliant way. To enhance application and document interoperability, the development of *validators*[25] for ODF and OOXML documents together with the provisioning of test suites comprising *test scenarios and documents*[26] is of the essence. The first projects in this context have already been launched and are currently in progress. [27]

There are still many unanswered questions in the field of document interoperability. However, current developments in the field of the standardization as well as interested communities in the Open Source environment are working on and providing solutions to solvable questions and providing clearer views on limitations. Thus, before choosing the format of the document and the tools, it is essential to be aware of the proper reasons why the exchange of documents is necessary and what requirements are needed for the translation.

[25] (Probatron, 2009), (FhI Fokus, 2009)
[26] (opendocsociety.org, 2009) (OASIS), (FhI Fokus, 2009)
[27] (DII, 2009)

7 References

Altova. XML Spy. *XML Editor, Data Management, UML and Web Services tools.* [Online] [Cited: 12 July 2009.] http://www.altova.com/.

DII. 2009. DII implementer's notes. *ODF 1.1.* [Online] June 2009. [Cited: 12 July 2009.] http://www.documentinteropinitiative.org/OASISODF1.1/reference.aspx.

Ditch, Walter. 2007. XML-based Office Documents. *JISC Technology and Standards Watch.* August 2007.

ECMA-376-1, Second Edition. 2008. *Office Open XML File Formats, Part 1: Fundamentals and Markup Language Reference.* December 2008.

ECMA-376-2, Second Edition. 2008. *Office Open XML Formats, Part 2: Open Packaging Conventions.* December 2008.

ECMA-376-3, Second Edition. 2008. *Office Open XML File Formats, Part 3: Markup Compatibility and Extensibility.* December 2008.

ECMA-376-4, Second Edition. 2008. *Office Open XML File Formats, Part 4: Transitional Migration Features.* December 2008.

FhI Fokus. 2009. OOXML Validator. [Online] July 2009. [Cited: 12 July 2009.] http://www.is29500.com/.

ISO. 1989. ODA/ODIF ISO 8613-1:1989. [Online] 1989. [Cited: 12 July 2009.] http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=15926.

—. 2006. *OpenDocument Format ODF.* 30 November 2006. ISO/IEC 26300:2006(E).

—. 1986. SGML ISO 8879:1986. [Online] ISO, 1986. [Cited: 12 July 2009.] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=16387.

Microsoft. 2008. Microsoft Expands List of Formats Supported in Microsoft Office 2010. [Online] 2008. [Cited: 12 July 2009.] http://www.microsoft.com/Presspass/press/2008/may08/05-21ExpandedFormatsPR.mspx.

—. 2006. Microsoft Open Specification Promise. [Online] 2006. [Cited: 12 July 2009.] http://www.microsoft.com/interop/osp/default.mspx.

—. 2007. Open XML Developer Workshop. [Online] 2007. [Cited: 12 July 2009.] http://openxmldeveloper.org/articles/DeveloperWorkshopContent.aspx.

OASIS. OpenDocument Fellowship. *Test Suite.* [Online] [Cited: 12 Juli 2009.] http://develop.opendocumentfellowship.com/testsuite/.

—. 2005. *OpenDocument v1.0 Specification.* http://www.oasis-open.org/committees/download.php/12572/OpenDocument-v1.0-os.pdf : s.n., May 2005.

—. 2007. *OpenDocument v1.1 Specification.* http://docs.oasis-open.org/office/v1.1/OS/OpenDocument-v1.1.pdf : s.n., February 2007.

—. 2009. *OpenDocument v1.2 Specification.* http://www.oasis-open.org/committees/download.php/32432/OpenDocument-v1.2-cd02.pdf : s.n., April 2009.

opendocsociety.org. 2009. ODF plugtest. [Online] June 2009. [Cited: 12 July 2009.] http://plugtest.opendocsociety.org/doku.php.

OpenOffice. 2002. The OpenOffice.org community announces the availability of OpenOffice.org 1.0. [Online] April 2002. [Cited: 12 July 2009.] http://www.openoffice.org/about_us/ooo_release.html.

Oxygen. Oxygen XML editor. [Online] [Cited: 12 July 2009.] http://www.oxygenxml.com/.

Probatron. 2009. Office-o-tron. *Office Document Validation.* [Online] June 2009. [Cited: 12 July 2009.] http://www.probatron.org:8080/officeotron/officeotron.html.

Schmidt, Kay-Uwe, et al. 2006. Document Interoperability for Use in eGovernment - Integration of XML-based Document Content in Public Administration Processes. [ed.] Fraunhofer Institute for Open Communication Systems FOKUS. *FOKUS White Paper.* May 2006.

Vugt, Wouter van. 2009. Open XML Package Explorer. [Online] June 2009. [Cited: 12 July 2009.] http://packageexplorer.codeplex.com.

W3C. 2006. Extensible Markup Language (XML) 1.1 (Second Edition). [Online] November 2006. [Cited: 12 July 2009.] http://www.w3.org/TR/2006/REC-xml11-20060816/.

—. 2003. Mathematical Markup Language (MathML) Version 2.0 (Second Edition). [Online] 21 October 2003. [Cited: 15 July 2009.] http://www.w3.org/TR/MathML2/.

—. 2003. XForm 1.0 (First edition). [Online] 14 October 2003. [Cited: 15 July 2009.] http://www.w3.org/TR/xforms/.

—. 2002. XHTML 1.0 The Extensible HyperText Markup Language (Second Edition). [Online] August 2002. [Cited: 12 July 2009.] http://www.w3.org/TR/xhtml1/.

Fraunhofer FOKUS

www.fokus.fraunhofer.de/go/egov-lab

Imprint

Publisher
Fraunhofer-Institute for
Open Communication Systems FOKUS
Kaiserin-Augusta-Allee 31
10589 Berlin, Germany

Authors
Dr. Klaus-Peter Eckert
Jan Henrik Ziesing
Ucheoma Ishionwu

Contact
Competence Center ELAN
Electronic Government and Applications
Phone +49 (0)30 3463 7115
eMail elankontakt@fokus.fraunhofer.de
www.fokus.fraunhofer.de/go/egov-lab