OASIS

The State of ODF Interoperability Version 1.0

Committee Specification 01

10 December 2010

Specification URIs:

This Version:
http://docs.oasis-open.org/oic/StateOfInterop/v1.0/cs01/StateOfInterop-v1.0-cs01.odt (Authoritative)
http://docs.oasis-open.org/oic/StateOfInterop/v1.0/cs01/StateOfInterop-v1.0-cs01.html
http://docs.oasis-open.org/oic/StateOfInterop/v1.0/cs01/StateOfInterop-v1.0-cs01.pdf

Previous Version:
http://docs.oasis-open.org/oic/StateOfInterop/v1.0/cd06/StateOfInterop-v1.0-cd06.odt (Authoritative)
http://docs.oasis-open.org/oic/StateOfInterop/v1.0/cd06/StateOfInterop-v1.0-cd06.html
http://docs.oasis-open.org/oic/StateOfInterop/v1.0/cd06/StateOfInterop-v1.0-cd06.pdf

Latest Version:
http://docs.oasis-open.org/oic/StateOfInterop/v1.0/StateOfInterop.odt (Authoritative)
http://docs.oasis-open.org/oic/StateOfInterop/v1.0/StateOfInterop.html
http://docs.oasis-open.org/oic/StateOfInterop/v1.0/StateOfInterop.pdf

Technical Committee:
OASIS Open Document Format Interoperability and Conformance (OIC) TC

Chair(s):
Bart Hanssens, Fedict

Editor(s):
Robert Weir, IBM

Related Work:

This specification is related to:

- OASIS Open Document Format

Abstract:
This report discusses interoperability with respect to the OASIS OpenDocument Format (ODF) and notes specific areas where implementors might focus in order to improve interoperability among ODF-supporting applications.

Status:

This document was last revised or approved by the ODF Interoperability and Conformance TC on the above date. The level of approval is also listed above. Check the "Latest Version" or "Latest Approved Version" location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at http://www.oasis-open.org/committees/oic/.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (http://www.oasis-open.org/committees/oic/ipr.php.

The non-normative errata page for this specification is located at http://www.oasis-open.org/committees/oic/.

Notices

Table of Contents

1 Introduction

OASIS OpenDocument Format (ODF) is a standard for office documents, including text documents, spreadsheets and presentations. ODF 1.0 was published in 2005, and ODF 1.1 was published in 2007. ODF 1.0 was also approved as ISO/IEC 26300:2006. The OASIS ODF Technical Committee[1] is currently working on ODF 1.2.

The OASIS ODF Interoperability and Conformance (OIC) TC was created in October 2008 with the stated purpose "to produce materials and host events that will help implementors create applications which conform the ODF standard and which are able to interoperate."

The charter of the OIC TC also calls for it to periodically review the state of conformance and interoperability among ODF implementations, to report on "overall trends in conformance and interoperability", to note "areas of accomplishment as well as areas needing improvement" and to "recommend prioritized activities for advancing the state of conformance and interoperability among ODF implementations."

This *State of ODF Interoperability* report is the first of the OIC TC's reports on interoperability, and as such provides an overview of the topic and discusses the baseline level of achievement. Future reports will focus on progress achieved beyond this baseline.

1.1 Normative References

This report contains no normative references.

1.2 Non-Normative References

This report contains no non-normative references. However, footnotes containing hyperlinks to relevant online resources are provided.

---

[1]http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=office

2 Conformance and Interoperability

2.1 ODF Conformance

Conformance is the relationship between a product[1] and a standard. A standard defines provisions that constrain the allowable attributes and behaviors of a conforming product. Some provisions define mandatory requirements, meaning requirements that all conforming products must satisfy, while other provisions define optional requirements, meaning that where applicable they must be satisfied. Conformance exists when the product meets all of the mandatory requirements defined by the standard, as well as those applicable optional requirements. For example, validity with respect to the ODF schema is a mandatory requirement that all conformant ODF documents must meet. However, support of the "fo:text-align" attribute is not required. Nevertheless, there are optional requirements that constrain how this feature must be implemented by products that do support that feature, namely that its value is restricted to "start", "end", "left", "right", "center" or "justify".

A standard may define requirements for one or more conformance *targets* in one or more *classes*, in which case a product may be said to conform to a particular conformance target and class. For example, ODF 1.1 defines requirements for an ODF Document target as well as an ODF Consumer target. ODF 1.2, in its current draft, defines requirements for an additional conformance class for the ODF Document target, namely the Extended ODF Document class.

There have been several attempts to assess conformance of ODF Documents. The ODF Fellowship hosts an online validator[2] that allows the user to upload a document which is then validated according to the ODF schema. Oracle hosts a similar tool at the OpenOffice.org web site[3]. The ODFToolkit Union has a ODF Conformance project[4] to further develop this code. Office-a-tron[5] is another validator. The ODF TC has posted instructions for how to validate ODF document instances using a variety of tools[6].

These tools primarily look at document validity, an XML concept, which is a necessary but insufficient condition for document conformance. These tools are limited to a static examination of document contents and are not capable of testing conformance of ODF applications. However, we believe that XML validation, combined with additional static analysis, is a promising approach to automate conformance testing of ODF documents.

Based on the results from existing online validators, which only test a subset of conformance requirements, the overall level of document conformance appears to be good. However, there are some recurring issues, including:

1. Style names beginning with a number, making them an invalid NCName. Implementations with this issue could trivially transfer their style names into valid ones by pre-pending an underscore character ('_') to the style name.

2. Documents missing a mimetype file

3. MathML documents with non-standard doctype declarations.

---

[1]'Product' here is used in its neutral sense, as "something produced". It is not intended to connote a commercial use.

[2]http://opendocumentfellowship.com/validator

[3]http://tools.services.openoffice.org/odfvalidator/

[4]http://odftoolkit.org/projects/odftoolkit/pages/ODFValidator

[5]http://www.probatron.org:8080/officeotron/officeotron.html

[6]http://wiki.oasis-open.org/office/How_to_Validate_an_ODF_document

Since automated validation testing of documents is a cheap and effective way to find errors in ODF documents, its widespread use is especially encouraged.

2.2 ODF Interoperability

According to ISO/IEC 2382-01, "Information Technology Vocabulary, Fundamental Terms", interoperability is "The capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units".

From the perspective of ODF, the document is the data which is transferred, and the functional units are the software applications which create, edit, view and manipulate these documents. Where the document can be successfully transferred among such applications, without the user needing to be concerned with the unique characteristics of each application, then interoperability is high. Conversely, where the user needs to be aware of the quirks of each application, there interoperability is poor.

Since the capabilities of ODF applications extend beyond the common desktop editors, and include other product categories such as web-based editors, mobile device editors, document convertors, content repositories, search engines, and other document-aware applications, interoperability will mean different things to users of these different applications. However, to one degree or another, interoperability consists of meeting user expectations regarding one or more of the following qualities when transferring documents:

1. The visual appearance of the document at various levels, e.g., glyph, run, line, block, page, etc.

2. The structure of the document as revealed when the user attempts to edit the document, e.g., headers, paragraphs, lists, tables.

3. The behaviors and capabilities of internal and external links and references.

4. The behaviors and capabilities of embedded images, media and other objects.

5. The preservation of document metadata.

6. The preservation of document extensions.

7. The integrity of digital signatures and other protection mechanisms.

8. The runtime behaviors manifest from scripts, macros and other forms of executable logic.

In any given user task, one or more of these qualities may be of overriding concern.

2.3 Conformance and Interoperability

The relationship between conformance and interoperability is subtle and often confused.   On one hand, it is possible to have good interoperability, even with non-conformant documents. Take HTML, for example. A study of 2.5 million web pages found that only 0.7% of them conformed to the HTML standard[1]. The other 99.3% of HTML pages were not conformant. So conformance is clearly not a prerequisite to interoperability. On the other hand, web browsers require significant additional complexity to handle the errors in non-conformant documents. This complexity comes at a tangible cost, in development resources required to write a robust (tolerant of non-conformance) web browser, and well as less tangible liabilities, such as greater code complexity which typically results in slower performance and decreased reliability. So although conformance is not required for interoperability, we observe that interoperability is most efficiently achieved in the presence of conforming applications and documents. However, this is an ideal alignment that is rarely achieved, since standards have defects, applications have bugs and users

---

[1] https://bora.uib.no/handle/1956/1731?mode=full

make mistakes. So, in practice, achieving a satisfactory degree of interoperability almost always requires additional efforts beyond mere conformance.

3 An Interoperability Model

3.1 Sources of Interoperability Problems

A model for describing document interoperability is given in Figure 1.
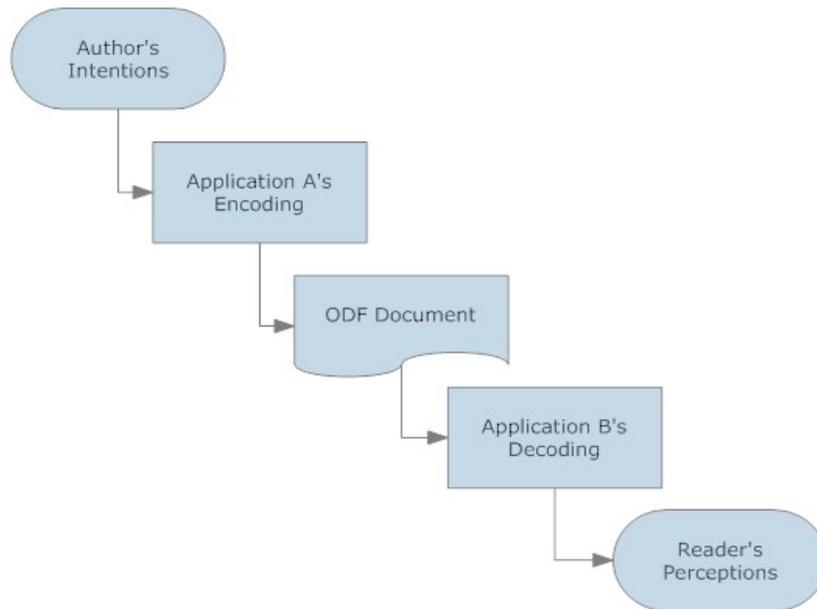
Figure 1: Interoperability Model

Any document exchange scenario will involve these steps:

1. Authoring: The human author of the document express his intentions by instructing a software application, typically via a combination of keyboard and GUI commands. Note that computer-generated documents also have an authoring step, in which case the human intentions are expressed by the author of the document generation software, via his code instructions.

2. Encoding: The software application, when directed to save the document, executes program logic to encode into ODF format a document that corresponds to the instructions given to it by the user.

3. Storage: The ODF document then represents a transferable data store that can be given to another user.

4. Decoding: The receiving user then uses their software application to decode the ODF document, and render it in fashion suitable for User B to interact with it.

5. Interpretation: User B then perceives the document in their software application and interprets the original author's intentions.

Interoperability defects can be introduced at any step in this process. For example:

1. An inexperienced user might instruct the authoring software application incorrectly, so that his instructions do not match his intentions. For example, a user may try to center text by padding a line with extra spaces rather than using the text align feature of their word processor. Or he might try to express a table header by merely applying the bold attribute to the text rather than defining it as a table header.

2. The software application that writes the document may have defects that cause it to incorrectly encode the user's instructions.

3. Due to ambiguities in the ODF specification, the document may be subject to different interpretations by Application A and Application B.

4. The software application that reads the document may have defects that cause it to incorrectly decode and render the document.

5. The user reading the document may incorrectly perceives its contents.

3.2 Round-trip Interoperability

Round-trip interoperability refers to scenarios where a document author collaborates with one or more other users, such that the document will be edited, not merely viewed, by those other users, and where the revised document is eventually returned to the original author.

From the perspective of the interoperability model, this introduces nothing new. A round-trip scenario is simply a iteration of the above steps, with the same opportunities for errors being introduced, e.g., A→B→A is the same as A→B, followed by B→A. However, since errors introduced at any step in the process tend to accumulate, a complex round-trip scenario will tend to suffer more in the presence of any interoperability defects.

Also, since the original author is the person who most knows the author's intentions, he will also be the most sensitive to the slightest alterations in content or formatting introduced into his document. So minor differences that might not have been noticed when read by a second user will be more obvious to the original author. This sensitivity to even the smallest differences will tend to cause the perception of round-trip interoperability to be lower.

StateOfInterop-v1.0-cs01

10 December 2010
Page 10 of 17

4 Problems and Solutions

4.1 Priority Areas for Improvement

Based on initial testing in the OASIS ODF Interoperability and Conformance TC, as well as scenario-based testing at the ODF Plugfest, several feature areas have been identifies as needing improvement in one or more ODF editors:

1. Some ODF features are not widely-implemented, such as document encryption, change tracking and form controls.

2. There are implementation bugs and inconsistencies that reduce interoperability of embedded charts, specifically:

   a) When writing out embedded charts, some implementations write out a link to the embedded chart asxlink:href="Object 1/". Other implementations write it out as: xlink:href="./Object 1".

   b) Some optional attributes, such as table:cell-range-address and chart:values-cell-range-address are not written by all implementations, although some implementations will not correctly render a chart without these values being set. More details are on the ODF Plugfest wiki.[1]

3. Implementations vary in their ability to to parse spreadsheet formulas written by other implementations. Adoption of OpenFormula (part of ODF 1.2) as the interchange format for spreadsheet formula expressions is encouraged. When an implementation does not support a formula syntax, typically the values are shown when office:value is present, but formulas themselves are removed. A simple example is a SUM function that calculates the sum of 2 cells. When opening a spreadsheet, a user may not notice any difference because the values are still there. But when changing the values of these 2 cells, the value of the cell originally containing the SUM function will not be updated accordingly because the formula has been removed.

4. Although simple lists appear to work well across the tested implementations, documents using advanced list structures (nested lists and list continuations) fared less well. When a document containing change tracking information is loaded in an implementation that does not support change tracking, the <text:tracked-changes> element and its content can be removed. Also, the content of deleted paragraphs, stored in <text:p> elements inside <text:tracked-changes>, can become visible at the top of the first page of the document (because <text:tracked-changes> is stored before all other <text:p>'s), which can be very confusing for the user

4.2 Approaches to Improving ODF Interoperability

Improving interoperability generally follows three steps:

1. Define the expected behavior

2. Identify defects in implementations

3. Fix the defects

The primary definition of expected behavior for the rendering of ODF documents is the published ODF standard. However, there are other sources of expected behavior, and meeting these expectations, where they do not conflict with the ODF standard, are, from the user's perspective, very important as well. For example:

---

[1] http://plugtest.opendocsociety.org/doku.php?id=scenarios:20090615:chartspread

1. Approved Errata[1] to the ODF standard

2. Other publications of the OASIS ODF TC, such as the Accessibility Guidelines[2]

3. Draft versions of ODF, where they clarify the expected behavior

4. Interoperability Guidelines, as may be published by the ODF Interoperability and Conformance TC.

5. Common sense and convention, which often provides a shared set of expectations between the user and the application vendor. For example, the ODF standard does not define the exact colorimetric value of the color "red", though undoubtedly users would be surprised to see their word processor render it as yellow.

There are several tools, processes and techniques that have been suggested for identifying interoperability defects, including:

6. Automated static testing of ODF documents, which could range in complexity from simple XML validation to more involved testing of conformance and portability.

7. "Atomic" conformance tests, which test the interoperability of individual features of the ODF standard at the lowest level of granularity. The ODF Fellowship's *OpenDocument Sample Documents*[3] is an early example of this approach. The approach used by Shah and Kesan[4] falls into this category as well.

8. Scenario-based testing, which combine multiple ODF features into documents which reflect typical real-world uses.

9. "Acid" tests aim to give the end-user a quick view how well their application supports the standard. This approach was popularized in the Web Standards Project[5] to improve browser interoperability. Sam Johnston has created a prototype ACID test for spreadsheet formulas[6].

10. Plugfests, face-to-face and virtual This approach has proven useful in interactive testing among vendors in the ODF Plugfests[7].

11. OfficeShots.org[8] allows an end-user to upload a document and compare how it will render in different applications. This can allow the user to see potential interoperability problems before they distribute their document.

12. User-submitted bug reports, sent to their application vendor, can help the vendor prioritize areas which are in need of improvement.

13. Public comments, submitted to the OASIS ODF TC[9], which report ambiguities in the specification which can impact interoperability. Such comments can be resolved in errata or revisions of the standard.

---

[1] http://docs.oasis-open.org/office/v1.0/errata/os/OpenDocument-v1.0-errata-01-os.html

[2] http://docs.oasis-open.org/office/office-accessibility/v1.0/cs01/ODF_Accessibility_Guidelines-v1.0.html

[3] http://testsuite.opendocumentfellowship.com/

[4] http://law.bepress.com/uiuclwps/papers/art97/

[5] http://www.webstandards.org/files/acid2/test.html#top

[6] http://sites.google.com/a/odfiic.org/acid/ods

[7] http://plugtest.opendocsociety.org/doku.php

[8] http://www.officeshots.org/

[9] http://www.oasis-open.org/committees/comments/index.php?wg_abbrev=office

As enumerated above, there are a variety of approaches to identifying interoperability defects. At this point it is not clear which of these techniques will, over the long term, be the most effective. Some require more substantial up-front investment in automation development, but this investment also permits a degree of test automation. Some approaches require substantial efforts in analysis of the ODF standard and construction of individual test cases. But once these test cases are designed, they can be executed many times at low cost. There are also techniques that require far less advance preparation and are suitable for formal and informal testing at face-to-face plugfests.

These options are familiar in the field of software quality assurance (SQA), and from that field we are taught to consider several factors:

- What is the "defect yield" of each testing approach? The defect yield is the number of defects found per unit of testing time, and is a measure of the efficiency of any given approach.

- What is the test coverage that can be achieved by any given approach? Test coverage would indicate what fraction of the features of ODF are tested by that approach.

- Once an initial investment is made, how expensive will it be to update test materials as new ODF versions and new application versions are released?

- How well does the testing approach prioritize the testing effort, so that the defects that most impact interoperability for the most number of users are found quickly?

The goal should be to identify an approach that finds the most number of defects with the least effort, with an emphasis on those defects that have the greatest real-world impact. It is well-known from SQA research that that the optimal approach often involves a blend of different complementary techniques.

Of course, interoperability will not improve merely by testing. The results of testing must feed forward into the vendors' development plans, so these defects are fixed and the fixes make it into the hands of end-users. Nothing improves until vendors change their code. Merely talking about the problem doesn't make it go away.

5 Conclusion

This *State of ODF Interoperability* report is a snapshot in time based on current ODF implementations, current interoperability activities and the current thoughts of the OASIS ODF Interoperability and Conformance TC. We intend to periodically reissue this report as conditions evolve, to note progress and remaining challenges.

6 Conformance

This report is informative and contains no normative clauses or references.

Appendix A. Acknowledgments

The following individuals have participated in the creation of this specification and are gratefully acknowledged

Participants:

- Jeremy Allison, Google Inc.
- Mathias Bauer, Oracle Corporation
- Michael Brauer, Oracle Corporation
- Alan Clark, Novell
- Xiaohong Dong, Beijing Redflag Chinese 2000 Software Co., Ltd.
- Pierre Ducroquet
- Bernd Eilers, Oracle Corporation
- Andreas J. Guelzow
- Dennis E. Hamilton
- Bart Hanssens, Fedict
- Donald Harbison, IBM
- Shuran Hua, Beijing Redflag Chinese 2000 Software Co., Ltd.
- Mingfei Jia, IBM
- Peter Junge, Beijing Redflag Chinese 2000 Software Co., Ltd.
- Doug Mahugh, Microsoft Corporation
- Eric Patterson, Microsoft Corporation
- Tom Rabon, Red Hat
- Daniel Rentz, Oracle Corporation
- Andrew Rist, Oracle Corporation
- Svante Schubert, Oracle Corporation
- Charles Schulz, Ars Aperta
- Jerry Smith, US Department of Defense (DoD)
- Patrick Strader, Microsoft Corporation
- Louis Suarez-Potts, Oracle Corporation
- Alex Wang, Beijing Sursen International Information Technology Co., Ltd.
- Robert Weir, IBM
- Thorsten Zachmann

Appendix B. Revision History

- CD 05 Rev 1 (2010-09-08) Added revision history appendix, non-normative references

- CD 05 (2010-07-12) Updated based on public review comments

- CD 04 (2010-02-10) Corrected error in description of chart URI

- CD 03 (2009-11-27) Fixed numbered lists.

- CD 02 (2010-11-02) Explain OIC TC in intro, more detail in "Priority Areas for Improvement", spelling corrections

- CD 01 (2009-10-20) Initial version

StateOfInterop-v1.0-cs01
10 December 2010
Page 17 of 17