OFFICE OPEN XML OVERVIEW

REVISED – SEPTEMBER 9, 2008

ECMA TC45
TOM NGO (NEXTPAGE), EDITOR

1 INTRODUCTION

Office Open XML is an open standard (ISO/IEC 29500, ECMA-376 Edition 2) for word-processing documents, presentations, and spreadsheets that can be freely implemented by multiple applications on multiple platforms. Its publication benefits organizations that intend to implement applications capable of using the format, commercial and governmental entities that procure such software, and educators or authors who teach about the format. Ultimately, all users enjoy the benefits of an XML standard for their documents, including stability, preservation, interoperability, and ongoing evolution.

This white paper summarizes Office Open XML. Read it to:

- Understand the purposes of Office Open XML and the structure of the Specification;
- Know its properties: how it addresses backward compatibility, preservation, extensibility, custom schemas, subsetting, multiple platforms, internationalization, and accessibility; and
- Learn how to follow the high-level structure of any Office Open XML file, and navigate quickly to any portion of the Specification from which you require further detail.

2 PURPOSES OF THE STANDARD

2.1 INTEROPERABILITY FOR OFFICE OPEN XML DOCUMENTS

The Office Open XML standard helps to bring the benefits of multi-vendor interoperability to the pre-existing corpus of word-processing documents, presentations, and spreadsheets that are encoded in binary formats defined by Microsoft Corporation. The goal of the standard is to ensure that a document written by a conforming producer of Office Open XML can be read by a conforming consumer of Office Open XML, and that the two agree on the semantics of that document. Beyond the scope of the standard, but complementary to it, is a translation from the binary formats to Office Open XML. (See "Publication of the Binary Formats and the Binary to Office Open XML Mapping," below.)

At the time of writing, more than 500 million users have generated documents in the binary formats, with estimates exceeding 40 billion documents with billions more being created each year. Therefore, a top priority in the design of Office Open XML was to maximize its ability to faithfully represent the information contained in the binary corpus, relegating to transitional status only content that would interfere with the primary goal of interoperability. To achieve this, the standardization process consisted of mirroring in XML the capabilities required to represent the existing corpus, extending them, providing detailed documentation, and enabling interoperability.

2.2 WHY AN OPEN STANDARD?

The original binary formats were created in an era when space was precious and parsing time severely impacted user experience. They were based on direct serialization of in-memory data structures used by Microsoft® Office® applications. Modern hardware, network, and standards infrastructure (especially XML) permit a new design that favors implementation by multiple vendors on multiple platforms, and allows for evolution.

Concurrently with those technological advances, markets have diversified to include a new range of applications not originally contemplated in the simple world of document editing programs. These new applications include ones that:

- Generate documents automatically from business data;
- Extract business data from documents and feed those data into business applications;
- Perform restricted tasks that operate on a small subset of a document, yet preserve editability;
- Provide accessibility for user populations with specialized needs, such as the blind; or
- Run on a variety of hardware, including mobile devices.

Perhaps the most profound issue is one of long-term preservation. We have learned to create exponentially increasing amounts of information. Yet we have been encoding that information using digital representations that are so deeply coupled with the programs that created them that after a decade or two, they routinely become extremely difficult to read without significant loss. Preserving the financial and intellectual investment in those documents (both existing and new) has become a pressing priority.

The emergence of these four forces – extremely broad adoption of the binary formats, technological advances, market forces that demand diverse applications, and the increasing difficulty of long-term preservation – have created an imperative to define an open XML format and migrate the billions of documents to it with as little loss as possible. Further, standardizing that open XML format and maintaining it over time create an environment in which any organization can safely rely on the ongoing stability of the specification, confident that further evolution will enjoy the checks and balances afforded by a standards process.

Various document standards and specifications exist; these include HTML, XHTML, PDF and its subsets, ODF, DocBook, DITA, and RTF. Like the numerous standards that represent bitmapped images, including TIFF/IT, TIFF/EP, JPEG 2000, and PNG, each was created for a different set of purposes. Office Open XML addresses the need for a standard that covers the features represented in the existing document corpus. To the best of our knowledge, it is the only XML document format that supports every feature in the binary formats.

3 STRUCTURE OF THE SPECIFICATION

Office Open XML is a multipart standard that defines formats for word-processing, presentation, and spreadsheet documents. Each type of document is specified through a primary markup language: WordprocessingML (WML), PresentationML (PML), or SpreadsheetML (SML). Embedding mechanisms permit a document of any one of these three types to contain material in the other primary markup languages and in a number of supporting markup languages.

The Specification contains both normative material (material that defines Office Open XML) and informative material (material that aids the reader's understanding but is not prescriptive). It is structured in Parts to meet the needs of varying audiences; for example, Part 4 documents those features aimed at backwards compatibility with existing binary documents.

Part 1 – Fundamentals and Markup Language Reference

- Defines vocabulary, notational conventions, and abbreviations.
- Summarizes the three primary markup languages and the supporting markup languages.
- Defines the conformance classes WML Strict, SML Strict, and PML Strict; establishes conditions for conformance and provides interoperability guidelines.
- Describes Office Open XML's use of the Open Packaging Conventions
- Defines every strict element and attribute, parent/child relationships for elements, and additional semantics.
- Declares the XML schemas for the markup languages as XML Schema Definitions (XSD) [2], in an annex that is also published electronically. The annex also expresses them non-normatively using RELAX NG (ISO/IEC 19757-2) [4].
- In an informative Primer annex, introduces the features of each markup language, providing context and illustrating elements through examples and diagrams.
- Describes the facility for storing custom XML data within a package to support integration with business data.
- Describes the differences between ISO/IEC 29500 (ECMA-376 Edition 2) and ECMA-376 Edition 1.

Part 2 – Open Packaging Conventions

- Defines the Open Packaging Conventions (OPC). Every Office Open XML file comprises a collection of byte streams called parts, combined into a container called a package. The packaging format is defined by the OPC.
- Describes a recommended physical implementation of the OPC that uses the Zip file format.
- Declares the XML schemas for the OPC as XML Schema Definitions (XSD) [2], in an annex that is also published electronically. The annex also expresses them non-normatively using RELAX NG (ISO/IEC 19757-2) [4].

Part 3 – Markup Compatibility and Extensibility

- Describes facilities for extension of Office Open XML documents.
- Specifies elements and attributes by which applications with different extensions can interoperate.
- Expresses extensibility rules using NVDL (ISO/IEC 19757-4) [5].

Part 4 – Transitional Migration Features

- Defines features that are for backward compatibility and are useful for high-quality migration of existing binary documents to ISO/IEC 29500.
- Defines the conformance classes WML Transitional, SML Transitional, and PML Transitional.
- Defines every transitional element and attribute, parent/child relationships for elements, and additional semantics.

In order to ease reading and navigation through these documents, the electronic versions have many internal active links. In particular, Part 1 has links to parent and child elements throughout.

4 PROPERTIES OF THE STANDARD

This section prepares you to investigate Office Open XML by describing some of its high-level properties. Each subsection describes one of these properties and refers to specific features within Office Open XML.

- "Interoperability" describes how Office Open XML is independent of proprietary formats, features, and run-time environment, allowing developers a broad range of choices.
- "Internationalization" mentions a few representative ways in which Office Open XML supports every major language group.
- "Low Barrier to Developer Adoption", "Compactness", and "Modularity" list specific ways in which Office Open XML avoids or removes practical impediments to implementation by diverse parties: learning curve, minimum feature set, and performance.
- "High Fidelity Migration" describes how Office Open XML meets the over-arching goal to preserve the information, including the original creator's full intent, in existing and new documents.
- "Integration with Business Data" describes how Office Open XML incorporates business information in custom schemas to enable integration and reuse of information between productivity applications and information systems.
- "Room for Innovation" describes how Office Open XML prepares for the future by defining further extensibility mechanisms and providing for interoperability between applications with differing feature sets.
- "Accessibility" describes how application developers and content creators can and should use Office Open XML to meet the needs of people with disabilities.

The remainder of this document, including this section, is a topical guide to Office Open XML. References to the ISO/IEC Standard have the form §*part*:*clause*.*subclause*; for example, §1:2.5 refers to Part 1, subclause 2.5 of the Standard. References to other headings within this paper are by name.

4.1 INTEROPERABILITY

Developers can write applications on multiple platforms that consume and produce Office Open XML.

Foremost, the interoperability of Office Open XML has been accomplished through extensive contributions, modifications, and review of the specification by members of the Ecma TC45 committee [1] with diverse backgrounds and corporate interests. Representation included:

- Vendors (Apple, Intel, Microsoft, NextPage, Novell, and Toshiba) with multiple operating systems (Linux, MacOS, and Windows) and multiple intended uses of Office Open XML;
- Corporations (BP, Barclays Capital, Essilor, Statoil) with heavy investments in existing content, including mission-critical transaction systems; and
- The British Library and the United States Library of Congress, both of which have direct interest in preservation.

Then, during the ISO/IEC fast-track process, ISO and IEC National Bodies performed a technical review of the specification. Through the work of hundreds of participants in 87 countries, representing a very wide range of users of the standard, the National Bodies submitted more than one thousand unique comments and questions, which were resolved prior to final approval.

Throughout the standardization process, committee members raised and resolved hundreds of issues regarding policy, clarity, semantics, and possible dependence on environment. Representative clusters of activity included:

- Features to support platform independence of mechanisms that were proprietary in the original binary formats;
- Conditions for conformance;
- Contents of the schemas;

- Alternate representations for the schemas and extensibility mechanisms using RELAX NG (ISO/IEC 19757-2) [4] and NVDL (ISO/IEC 19757-4) [5];
- Representation of calendar dates;
- Internationalization, e.g., language codes, flexibility in defining work days and weekends, and richer information to support right-to-left and left-to-right scripts; and
- Completeness, correctness, and clarity of descriptions throughout the specification, in many cases as a result of attempting to implement portions of the specification.

The remainder of this subsection highlights specific areas in which Office Open XML departs from the original binary formats for the sake of interoperability.

One of the central requirements for interoperability is independence from any particular type of source content.

- Office Open XML contains no restriction on image, audio or video types. For example, images can be in GIF, PNG, TIFF, PICT, JPEG or any other image type (§1:15.2.14).
- Embedded controls can be of any type, such as Java or ActiveX (§1:15.2.9).
- WordprocessingML font specifications can include font metrics and PANOSE [17] information to assist in finding a substitution font if the original is not available (§1:M.1.9.5).

In addition, Office Open XML avoids dependence on the run-time environment of the application that produced a document.

- The classic example occurs with an external control or application that generates an image for a portion of the display surface. To guard against the case in which the control or application is unavailable or cannot run in a given run-time environment, the document file can contain an image representation. This mechanism exists in the older binary formats as well.
- Office Open XML introduces a more general mechanism called the Alternate Content Block (§1:M.1.18.4), which can be used in various situations where a consuming application might not be capable of interpreting what a producing application wrote. It is used commonly in the context of extensibility. This mechanism is described further in the subsection "Room for Innovation".
- Office Open XML avoids dependence on any parameter that is meaningful in a document producer's environment, but may not be in the consumer's environment. For example, the parameter CT_SYSCOLOR is an index into a color table in the producing environment. To support portability to a different consuming environment, PresentationML allows the producer to cache the system color that was in use at the time that a document was created.

Finally and most fundamentally, Office Open XML conforms to open W3C standards such as XML [6] and XML Namespaces [7]. This fact alone allows a base level of interoperability across all platforms and operating systems that adhere to these open standards.

4.2 INTERNATIONALIZATION

Office Open XML supports internationalization features required by such diverse languages as Arabic, Chinese (three variants), French, Hebrew, Hindi, Japanese, Korean, Russian, and Turkish.

Office Open XML inherently supports Unicode because it is XML. In addition, Office Open XML has a rich set of internationalization features that have been refined over the course of many years. This list is representative.

Text orientation: Office Open XML supports left-to-right (LTR) and right-to-left (RTL) languages. It also supports bidirectional ("BiDi") languages such as Arabic, Farsi, Urdu, Hebrew, and Yiddish, which run from right to left but can contain embedded segments of text that runs left to right. In WordprocessingML, text direction can be controlled on both the paragraph level (§1:17.3.1.6) and the level of a run within a paragraph (§1:17.3.2.30). Similarly, in DrawingML text, it can be controlled on the

body level (§1:21.1.2.1.1), on the paragraph level (§1:21.1.2.2), and within numbered and bulleted lists. An informative annex, Part 1 Annex J, "Bidirectional Support", provides an overview. It includes guidance regarding use of the Unicode Bidirectional Algorithm.

Text flow: In WordprocessingML, the direction of text flow can be controlled at the level of a section or a table (§1:17.3.1.41) or at the level of a paragraph (§1:17.3.2.30). At the section and table levels, text flow can be controlled in the vertical and horizontal directions. This allows Office Open XML to support all potential text layouts (e.g., vertical lines flowing top to bottom and stacked left to right, to support Mongolian). This affects the layout of lists, tables, and other presentation elements. DrawingML also utilizes Kumimoji settings at the paragraph and run levels to flow text horizontally and numbers vertically (§1:21.1.2.2.3, §1:21.1.2.3.9). In WordprocessingML (§1:17.3.1.16) and PresentationML (§1:19.2.1.17), character flow can also be controlled using Kinsoku settings to specify which characters are allowed to begin and end a line of text.

Number representation: For field formatting in WordprocessingML (§1:17.16.4.3.1), paragraph/list numbering in WordprocessingML (§1:17.9), page numbering in WordprocessingML (§1:17.6.12) and numbering in DrawingML (§1:20.1.10.61), numbers can be formatted using any of several dozen number formats (§1:17.18.59), including Hiragana, Arabic, Abjad, Thai (and Thai Baht), cardinal text (e.g., "one hundred twenty-three"), Chinese, Korean (Chosung or Ganada), Hebrew, Hindi, Japanese, Roman, or Vietnamese. These facilities also support arbitrary radix-point values (e.g., "1.00" vs. "1,00") and list separators. Internationalized number formatting is particularly robust in SpreadsheetML, which supports all of those features in the cell formats (§1:18.8.30) and in references to external data (§1:18.13.12).

Date representation: In WordprocessingML and SpreadsheetML, the display format for a calendar date can be specified as Gregorian (six display variants), Hebrew, Hijri, Japanese (Emperor Era), Korean (Tangun Era), Saka, Taiwanese, or Thai. In a strictly conforming SpreadsheetML document, all dates are stored as ISO 8601-formatted strings (§1:18.17.4). Since many existing spreadsheets store dates and times as serial numeric values, instructions for conversion are provided (§1:18.17.4.1, §1:18.17.4.2, §1:18.17.4.3).

Formulas: The formula specification in SpreadsheetML provides several text conversion functions appropriate for different languages. Examples are BAHTTEXT (§1:18.17.7.22), and ASC (§1:18.17.7.11). The CONVERT function (§1:18.17.7.48) distinguishes between units of measure that have the same name for different measures in different parts of the world (e.g., cup, tablespoon).

Language identifiers: In WordprocessingML (§1:17.3.2.20) and DrawingML (§1:21.1.2.3), every paragraph and run can be tagged with a language identifier, allowing an application to select appropriate proofing tools and other language-specific functionality. Language identifiers are as defined by RFC 4646/BCP 47 [16]. Part 4, "Transitional Migration Features", provides a conversion table (§4:9.9.2) for legacy numeric language codes to codes compliant with BCP 47. In addition to an identifier for each language, Office Open XML supports the naming of a character set, a font family and a PANOSE value to aid the application in choosing an appropriate substitute set of characters when local support is not present.

4.3 Low Barrier to Developer Adoption

A developer can begin to write simple Office Open XML conforming applications within a few hours of beginning to read the Specification.

Although the Specification describes a large feature set, an Office Open XML conforming application need not support all of features in the Specification. The Conformance statement (§1:2) requires merely that a conforming consumer "not reject any conforming documents of at least one document conformance class" and that a conforming producer "be able to produce conforming documents of at least one document conformance class" (§1:2.5). It also provides Interoperability Guidelines that state the role of element semantics (§1:2.7).

A conforming application can have extremely focused functionality and is not required to implement all the functionality found in the standard. For example, it could be a batch processor that merely updates the copyright notices in a collection of word-

processing documents, or a text-to-speech reader that understands enough of a slide presentation to render its content in audio as the user navigates slide by slide. The structure of the file format allows such programs to be written with minimal knowledge of Office Open XML. Specifically:

- The file format conforms to well-established standards, especially XML and ZIP, for which mature tools exist.
- The file format makes use of the Open Packaging Conventions, which combine XML and ZIP with standard mechanisms to express relationships within a file. Because of this, a file's contents can often be navigated without knowledge of the tag semantics for any of the primary or supporting markup languages in Office Open XML.
- Elements deep in the XML tree can be accessed and modified without disturbing the rest of the structure.

Small details throughout the file formats, some of which were not present in the binary formats, support applications with minimal functionality by providing cached values. For example:

- Without implementing a paginator, an application such as a reader for the blind could offer page navigation using last-calculated page breaks (§1:17.3.3.13).
- Without implementing formulas and integrating with an external data source, a spreadsheet program could work from cached calculations (§1:M.2.9) and cached external data (§1:18.10.1.76).

A minimal conforming document is extremely simple; see the subsection "Minimal WordprocessingML Document".

4.4 COMPACTNESS

The Office Open XML file format supports the creation of high-performance applications. In this subsection, we describe some of the design points that result in a compact file, thereby speeding handling and parsing. In the next subsection, we show how modular file structure enables an application to accomplish many tasks by parsing or modifying only a small subset of a document.

An Office Open XML file is conventionally stored as a ZIP archive for purposes of packaging and compression, following the recommended implementation of the Open Packaging Conventions. Perhaps surprisingly, Office Open XML files are on average 25% smaller, and at times up to 75% smaller, than their binary counterparts. For example, this white paper is 85% larger in the binary format!

A second simple source of compactness, particularly where an uncompressed representation is required, is the length of identifiers in the XML. Frequently used tag names are short. Implementers are encouraged to use short namespace prefixes as well; for example, the conventional prefix for the WordprocessingML namespace is "w".

Further compactness is achieved by avoiding repetition throughout the file format. In one class of examples, the goal is to avoid redundant storage of frequently reused objects.

- In SpreadsheetML, repeated strings are stored in a string table in the workbook, and referenced by index (§1:M.2.3).
- In SpreadsheetML, a formula that is filled down or across several cells is stored as a single "master" formula in the top left cell; the other cells in the fill range refer to it by a grouping index (§1:2.2.9.2).
- In DrawingML, shape names (§1:20.1.10.56), text geometries (§1:20.1.10.76), and other presets (several throughout §1.M.4.8, §1:M.4.9, and §1:20.1.10) are represented by name or number instead of explicitly. In these cases, the meanings of names and numbers reside in the Specification and not in the file. Here, the chosen representation is the result of an explicit tradeoff decision during the standards process. It is compact and allows editing at the correct level of abstraction: for example, a rectangle could be changed to an oval by changing one attribute (§1:20.1.9.18).

In another class of examples, hierarchy is used to provide inheritance semantics for editability. As a happy by-product, this increases performance by reducing file sizes.

- In WordprocessingML, styles are hierarchical (§1:M.1.8.9).
- In DrawingML, shapes are grouped hierarchically (§1:20.1.2.2.20).
- In PresentationML, a default hierarchy relates slide masters, slide layouts, and slides (§1:M.3.2).

Other aspects of Office Open XML are also designed to enable efficient implementation. For instance, in SpreadsheetML, the cell table stores only non-empty cells, and is capable of representing merged cells as a unit. The economy afforded by this technique is significant for sparse spreadsheets.

4.5 MODULARITY

An application can accomplish many tasks by parsing or modifying a small subset of the document.

Three features of the Office Open XML format cooperate to provide this modularity.

- A document is not monolithic; it is made of multiple parts.
- Relationships between parts are themselves stored in parts.
- The ZIP archive format that is typically used to support Office Open XML documents supports random access to each part.

For example:

- An application could move a slide cleanly from one presentation to another, together with resources such as images and layouts, entirely without parsing slide content. This example uses data called *explicit relationships* to find the slide and its resources. Explicit relationships are defined by the Open Packaging Conventions and can be parsed without any knowledge of PresentationML tag semantics (§1:9.2, §2:8.3).
- An application could strip all of the comments from a WordprocessingML document without parsing any of its contents (§1:11.3.2). This example uses data called *implicit relationships* to find the comments. Implicit relationships are specific to Office Open XML and therefore do require some knowledge of the relevant markup language (§1:9.2).

4.6 HIGH FIDELITY MIGRATION

Office Open XML is designed to support all of the features in the Microsoft Office 97-2003 binary formats.

It is difficult to overstate the difficulty of accomplishing this goal, and the consequent uniqueness of Office Open XML in doing so. Some formats, such as PDF, are designed to deliver a visual facsimile of a finished document to an end user. In contrast, Office Open XML is intended to permit future editing or manipulation at the same level of abstraction available to the original creator; for example, reducing a vector graphic to a bitmap would fall short of this intent, as would collapsing a style hierarchy to independent styles. Further, a document can contain computational semantics that the original creator expects to preserve, such as formula logic that depends on intermediate calculation results, including error codes or animation rules that produce dynamic behavior.

These references to the specification exemplify the ability of Office Open XML to represent subtle aspects of the binary formats.

- The SpreadsheetML description includes an extensive formula specification (§4:3.17.7).
- The WordprocessingML specification documents the rules by which paragraph, character, numbering, and table properties are composed with direct formatting (§1:M.1.8, especially §1:M.1.8.10).
- The PresentationML specification documents the animation features (§1:M.3.4).

Office Open XML enables multiple implementations to conform without having to match in every inconsequential detail. This is particularly important where numerical computations are involved, such as layout, effect rendering, and formula evaluation.

Requiring more consistency than is practical would create an unnecessarily high barrier for developers to achieve conformance. These statements underscore sample decisions made by the committee in this regard.

- Office Open XML defines effects such as surface appearances (§1:20.1.10.50) without constraining a developer to match those effects pixel for pixel.
- Office Open XML defines parameters such as page margins (§1:17.6.11), font (§1:17.8), and justification (§1:17.3.1.13). It allows developers to implement different flow algorithms as long as they respect those parameters.
- The SpreadsheetML formula specification (§1:18.17.7) does not attempt to remove variations in floating-point computation because, in general, doing so would require conforming applications to implement slow emulation instead of relying on native hardware. Instead, it specifies the minimum number of bits of precision for numerical calculations (§1:18.17.5).
- The SpreadsheetML formula specification also leaves certain conditional decisions implementation-defined, in order to allow for future innovation. For example, it does not limit how many times a computation such as NORMINV (§1:18.7.7.230) should iterate. (The NORMINV function performs the inverse of the normal distribution by performing an iterative search.)

A number of older features, such as VML (§1:M.5), are defined as transitional features. They are included only for backward compatibility. The use of newer standards already in Office Open XML, such as DrawingML (§1:M.4), is compulsory when writing new documents.

For further information regarding migration, see "Publication of the Binary Formats and the Binary to Office Open XML Mapping."

4.7 Integration with Business Data

Office Open XML enables organizations to integrate productivity applications with information systems that manage business processes. It does so through the use of custom schemas within Office Open XML documents. The goal is to reuse and to automate the processing of business information that is otherwise buried opaquely inside documents, where business applications cannot read or write it.

Applications include:

- *Search*: An end user could search a collection of spreadsheets for companies with profit margins exceeding 20%.
- *Metadata tagging*: A firm could tag presentations that have been approved from a regulatory perspective.
- *Document assembly*: A proposal group could streamline proposal generation by automating the preparation of the underlying data.
- *Data reuse*: A sales executive could generate a report of all sales contracts in a given date range, listing customers, deal sizes, and any modified terms and conditions.
- *Line-of-business applications*: Professionals in a specialized vertical could prepare deliverables in a familiar authoring environment, yet have their work products flow automatically into business systems.

Accomplishing these goals requires defining the structure and the type of data that a class of documents can contain, and allowing the information to be revealed wherever it occurs naturally within the flow of each document. Consider the simple example of a résumé. One would define a data structure that includes fields called name, phone number, address, career goals, and qualifications. One would then arrange for those fields to appear wherever human authors happen to put them in a document. In a different business setting, such as a finance group or a medical center, the structure and the data fields would be different.

Office Open XML allows this process to occur in a standardized fashion.

First, the structure of the business data is first expressed using a custom XML schema. This allows an organization to express data with tags that are meaningful from a business perspective. An organization can create its own schemas (§1:23.2.1), or use industry standard schemas such as XBRL for financial reporting [8] and HL7 for health-care information [9]. Schemas are being created in the public sector, inside corporations, and as industry standards, for purposes ranging from birth certificates to insurance information. Any custom schema can be used [2].

Second, the custom data are embedded in any Office Open XML document in a Custom XML part (§1:23) and can be described using a Custom XML Data Properties part (§1:22.5). By separating these custom data from presentation, Office Open XML enables clean data integration, while enabling end-user presentation and manipulation within a wide variety of contexts, including documents, forms, slides, and spreadsheets. Interoperability can thus be achieved at a more fundamental and semantically accurate level.

4.8 ROOM FOR INNOVATION

Office Open XML is designed to encourage developers to create new conforming applications that were not contemplated when the binary formats were defined, or even when Office Open XML was defined.

First, we discuss extensibility mechanisms that work together to allow interoperability between applications with differing feature sets. Consider an *up-level* application (one that contains a new feature not documented in Office Open XML) and a *down-level* application (one that does not understand that feature). The three primary goals of extensibility are:

- *Visual fidelity*: the ability for the down-level application to display what the up-level application would display. This inherently requires that a file store multiple representations of the same data.
- *Editability*: the ability to edit one or more of the representations.
- *Privacy*: the ability to ensure that old versions of one representation do not remain after editing another representation, unexpectedly leaving information that a user believes is deleted or modified. An application can achieve this by eliminating or synchronizing representations.

A developer wishing to extend the Office Open XML feature set has two main options:

- Alternate content blocks: An alternate content block (§1:M.1.18.4 and §3:10.2) stores multiple representations of the same content, each within its own choice block. A down-level application reads one choice block that it is capable of reading. Upon editing, it writes as many choice blocks as it is capable of writing.
- Extension lists: An extension list (§1:M.6) stores arbitrary custom XML without a visual representation.

Developers have room to innovate outside of those extensibility mechanisms.

- *Alternate interaction paradigms*. Office Open XML specifies more than document syntax but less than application behavior. As described in the Conformance statement, it focuses on semantics (§1:2.2, §1.2.3). Consequently, a conforming application is free to communicate with an end user through a variety of means, or not communicate with an end user at all – as long as it respects the specified semantics.
- *Diverse computing environments*. The Conformance statement admits applications that can run on low or specialized capacity hand-held devices, including devices that support the disabled, where applications may implement only a subset of Office Open XML (§1:2.6). The Additional Characteristics mechanism permits a producing application to communicate its capacity limits (§1:M.7.1).

As indicated in the previous subsection, some of the most substantial opportunities for innovation do not involve rendering documents for direct user interaction. Instead, they involve machine-to-machine processing using XML message formats, e.g., via XML Web Services [10]. Although such applications have no user-visible behavior other than their operations on data

contained within Office Open XML documents, they are subject to document conformance (§1:2.4), which is purely syntactic, and application conformance (§1:2.5), which incorporates both syntax and semantics.

While it is impossible to enumerate all possible use cases for customized XML processing, one may anticipate XML-centric services that process Office Open XML documents for automatic extraction and insertion of custom data, custom security services such as XML Digital Signature (10) or XML Encryption [12], or even arbitrary XSLT transformations [13] that convert to and from other XML formats. Office Open XML places no prohibitions or limitations on such processing.

4.9 ACCESSIBILITY

Application developers and content creators who use Office Open XML are strongly urged to support users who have a variety of special needs (§1:K.1).

Annex K of Part 1 in the Specification provides detailed accessibility guidelines with in-depth, cross-technology examples that are applicable to broadly distributed software or content, and more specialized implementations of Assistive Technology (AT). In addition, the Office Open XML format itself supports compliance with those guidelines by providing alternate content fields, tags to describe alternate input modalities, and a representation that makes alternate renderings simple to develop. The following examples, which are taken from Annex K, highlight a few these points.

- *Using elements designed for alternate input modalities*. In an accessible application, a user should be able to do everything using only the keyboard (§1:K.3.2). Elements such as `tabIndex` (§1:K.4.6) support this guideline.
- *Streamlining navigation via generic semantic structures*. Moreover, the interaction should be reasonably efficient; a user should not have to tab through hundreds of fields to reach a desired target (§1:K.3.2). Hierarchical relationships, for example those relating content controls (§1:K.4.7), ease navigation.
- *Exposing hierarchy*. Hierarchical relationships are useful for more than navigation; they are often essential to the meaning of a document. For example, complex tables can contain significant visual hierarchy without being explicitly nested. Office Open XML supports multi-level headers, giving developers in the AT community a way to expose that visual hierarchy using a variety of alternate renderings (§1:K.4.5).
- *Exploiting the low barrier to developer adoption*: Alternate renderings of Office Open XML documents should always reveal the semantic structure of a document. Even small developers can quickly develop meaningful applications that do so. For example, an audio slide reader can use explicit relationships (§1:9.2) to identify slides and their titles for accessible navigation (§1:K.4.4) without having to parse slide content.
- *Embedding alternate content*: Alternate content is especially important for rich media. For example, the recommended approach for handling text equivalents for animation is to create a separate text box for each animation. Content creators are encouraged to use the description attributes in Office Open XML for this purpose (§1:K.5.3), making it possible for developers to render text equivalents that are meaningful to the viewer.

Developers are also encouraged to maximize benefits to users with special needs by providing bridges to existing assistive technologies. Because Office Open XML is an open, standardized format, and because it is capable of representing the information necessary to support accessibility, it is well suited for conversion to other widely used formats. For example, an open-source effort has already produced a translator from Office Open XML to DAISY (Digital Accessible Information System, ANSI/NISO Z39.86-2005); see http://sourceforge.net/projects/openxml-daisy.

5 STRUCTURE OF AN OFFICE OPEN XML DOCUMENT

A primary objective of this white paper is to enable the reader to follow the high-level structure of any Office Open XML file. To accomplish this, we provide a moderate level of detail regarding the Open Packaging Conventions (OPC), and less detail regarding the individual markup languages.

5.1 OPEN PACKAGING CONVENTIONS

The Open Packaging Conventions (OPC) provide a way to store multiple types of content (e.g., XML, images, and metadata) in a container, such as a ZIP archive, to fully represent a document. They describe a logical model for representing containment and relationships.

The recommended implementation for the OPC uses the ZIP archive format. One can inspect the structure of any Office Open XML file by using any ZIP viewer. It is useful to inspect the contents of a small Office Open XML file in this manner while reading this description. On the Windows operating system, one needs only to add a ".zip" extension to the filename and double-click.

Logically, an Office Open XML document is an OPC *package* (§2:8). A package is a flat collection of *parts* (§2:8.1). Each part has a case-insensitive *part name* that consists of a slash-delimited sequence of segment names such as "/pres/slides/slide1.xml" (§2:8.1.1). Each part also has a *content type* (§2:8.1.2). Physically, the ZIP archive is one package, each ZIP item in the archive is one part, and pathnames within the ZIP archive correspond directly to part names.

In the ZIP implementation, "/[Content_Types].xml" allows a consumer to determine the content type of every part in the package (§2:9.2.6). The syntax and definition of media types follows section 3.7 of RFC 2616 [14].

Packages and parts can contain *explicit relationships* (§1:9.2) to other parts within the package, as well as to external resources. Every explicit relationship has a relationship ID, which allows a part's content to refer to it; and a type, which allows an application to decide how to process it. Relationship types are named using URIs, enabling non-coordinating parties to safely create new types without conflict.

The set of explicit relationships for a given source package or part is stored in a *relationships part*. The relationships part for the package as a whole is called "/_rels/.rels"; the relationships part for a part called "/a/b/c.xml" is called "/a/b/_rels/c.xml.rels". The relationships parts (and, in the ZIP implementation, the content-type part) are the only specially named parts in a package. To open a package, an application must parse the package-relationships part and follow the relationships of appropriate type.

All other parts in an Office Open XML document hold Office Open XML, custom XML, or content of arbitrary type such as multimedia objects. The ability of a part to hold custom XML is a particularly powerful mechanism for embedding business data and metadata.

5.2 WORDPROCESSINGML

A WordprocessingML document is composed of a collection of *stories* (§1:M.1.1). Each story is one of the following: the main document (§1:M.1.3), the glossary document (§1:M.1.13), a subdocument (§1:M.1.18.2), a header (§1:M.1.11.1), a footer (§1:M.1.11.2), a comment (§1:M.1.14.5), a frame, a text box (§1:M.1.18.1), a footnote (§1:M.1.12.1), or an endnote (§1:M.1.12.2).

The only required story is the main document. It is the target of the package relationship whose type is:

http://schemas.openxmlformats.org/officeDocument/2006/relationships/officeDocument

A typical path from root to leaf in the XML tree would comprise these XML elements (§1:M.1.2):

- `document` – the root element of the main document (§1:M.1.3).
- `body` – body (§1:M.7.1). Can contain multiple paragraphs. Can also contain section properties specified in a `sectPr` element.
- `p` – paragraph (§1:M.1.4.1). Can contain one or more runs. Can also contain paragraph properties specified in a `pPr` element, which in turn can contain default run properties (also referred to as character properties) specified in a `rPr` element (§1:M.1.4.4).
- `r` – run (§1:M.1.4.2). Can contain multiple types of run content, primarily text ranges. Can also contain run properties (`rPr`). The run is a fundamental concept within Office Open XML. A run is a contiguous piece of text with identical properties; a run contains no additional text markup. For example, if a sentence were to contain the words "this is **three** runs", then it would be represented by at least three runs: "this is ", "**three**", and " runs". In this respect, Office Open XML differs significantly from formats that allow for arbitrary nesting of properties, such as HTML.
- `t` – text range (§1:M.1.4.3.1). Contains an arbitrary amount of text with no formatting, line breaks, tables, graphics, or other non-text material. The formatting for the text is inherited from the run properties and the paragraph properties. This element often uses the `xml:space="preserve"` attribute.

In this subsection, we have touched upon direct formatting of text by specifying paragraph and run properties. Direct formatting falls at the end of an order of application that also includes character, paragraph, numbering, and table styles, as well as document defaults (§1:M.1.8.10). Those styles are themselves organized into inheritance hierarchies (§1:M.1.8.9).

The subsection "Minimal WordprocessingML Document" below lists a WordprocessingML document in full.

5.3 PRESENTATIONML

A PresentationML document is described by a presentation part. The presentation part is the target of the package relationship whose type is:

> http://schemas.openxmlformats.org/officeDocument/2006/relationships/officeDocument

The presentation refers to these primary constructs (§1:M.3.4.2), which we list from top to bottom in the default hierarchy:

- slide masters, notes masters, and handout masters (§1:M.3.2.2), all of which inherit properties from presentation;
- slide layouts (§1:M.3.2.5), which inherit properties from slide master; and
- slides (§1:M.3.2.3) and notes pages (§1:M.3.2.4), which inherit properties from slide layouts and notes masters respectively.

Each master, layout, and slide is stored in its own part. The name of each part is specified in the relationship part for the presentation part. Each of the six parts other than presentation is structured in essentially the same way. A typical path from root to leaf in the XML tree would comprise these XML elements (§1:M.1.2):

- `sld`, `sldLayout`, `sldMaster`, `notes`, `notesMaster`, or `handoutMaster` – the root element.
- `cSld` – slide (§1:19.3.1.16). Can contain DrawingML elements (as described in the next two bullets) and other structural elements (as described below).
- `spTree` – shape tree (§1:19.3.1.45). Can contain group shape properties in a `grpSpPr` element (§1:19.3.1.23) and non-visual group shape properties in an `nvGrpSpPr` element (§1:19.3.1.31). This node and its descendants are all DrawingML elements. We list some DrawingML elements here because of their pivotal role in PresentationML.
- `sp` – shape (§1:19.3.1.43). Can contain shape properties in a `spPr` element (§1:19.3.1.44) and non-visual shape properties in an `nvSpPr` element (§1:19.3.1.34).

In addition to the DrawingML shape content, a `cSld` can contain other structural elements, depending on the root element in which it resides, as summarized in this table:

	Slide	Slide Layout	Slide Master	Handout Master	Notes Master	Notes Page
Common Data	X	X	X	X	X	X
Transition	X	X	X			
Timing	X	X	X			
Headers and Footers		X	X	X	X	
Matching Name		X				
Layout Type		X				
Preserve		X	X			
Layout List			X			
Text Style			X			

Properties specified by objects lower in the default hierarchy (slide master, slide layout, slide) override the corresponding properties specified by objects higher in the hierarchy. For example, if a transition is not specified for a slide, then it is taken from the slide layout; if it is not specified there, then it is taken from the slide master.

5.4 SpreadsheetML

A SpreadsheetML document is described at the top level by a workbook part. The workbook part is the target of the package relationship whose type is:

> http://schemas.openxmlformats.org/officeDocument/2006/relationships/officeDocument

The workbook part stores information about the workbook and its structure, such as file version, creating application, and password to modify. Logically, the workbook contains one or more sheets (§1:M.2.2); physically, each sheet is stored in its own part and is referenced in the usual manner from the workbook part. Each sheet can be a worksheet, a chart sheet, or a dialog sheet. We will discuss only the worksheet, which is the most common type. Within a worksheet object, a typical path from root to leaf in the XML tree would comprise these XML elements:

- `worksheet` – the root element in a worksheet (§1:M.2.2).
- `sheetData` – the cell table, which represents every non-empty cell in the worksheet (§1:M.2.2.4).
- `row` – one row of cells in the cell table (§1:M.2.2.8).
- `c` – one cell (§1:M.2.2.9). The `r` attribute indicates the cell's location using A1-style coordinates. The cell can also have a style identifier (attribute `s`) and a data type (attribute `t`).
- `v` and `f` – the value (§1:M.2.2.9.1) and optional formula (§1:M.2.2.9.2) of the cell. If a cell has a formula, then the value is the result of the most recent calculation.

Both strings and formulas are stored in shared tables (§1:M.2.3 and §1:M.2.2.9.2.1) to avoid redundant storage and speed loads and saves.

5.5 Supporting Markup Languages

Several supporting markup languages can also be used to describe the content of an Office Open XML document.

- DrawingML (§1:M.4) – used to represent shapes and other graphically rendered objects within a document.
- VML (§1:M.5) – a format for vector graphics that has been replaced by DrawingML and is included as part of transitional features for backwards compatibility.

- Shared MLs: Math (§1:M.6.1), Metadata (§1:M.6.2), Custom XML (§1:M.6.3), and Bibliography (§1:M.6.1.4).

5.6 MINIMAL WORDPROCESSINGML DOCUMENT

This subsection contains a minimal WordprocessingML document that comprises three parts.

The content-type part "/[Content_Types].xml" describes the content types of the two other required parts.

```
<Types xmlns="http://schemas.openxmlformats.org/package/2006/content-types">
    <Default Extension="rels"
             ContentType="application/vnd.openxmlformats-package.relationships+xml"/>
    <Default Extension="xml"
             ContentType="application/vnd.openxmlformats-officedocument.wordprocessingml.document.main+xml"/>
</Types>
```

The package-relationship part "/_rels/.rels" describes the relationship between the package and the main document part.

```
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
    <Relationship Id="rId1"
                  Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/officeDocument"
                  Target="document.xml"/>
</Relationships>
```

The document part, in this case "/document.xml", contains the document content.

```
<w:document xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main">
    <w:body>
        <w:p>
            <w:r>
                <w:t>Hello, world.</w:t>
            </w:r>
        </w:p>
    </w:body>
</w:document>
```

The Specification provides minimal documents and additional detail for WordprocessingML (§1:11.2), PresentationML (§1:13.2), and SpreadsheetML (§1:12.2).

6 HISTORY

The work to standardize Office Open XML started in December 2005 in Ecma International via its Technical Committee 45 (TC45), which included representatives from Apple, Barclays Capital, BP, The British Library, Essilor, Intel, Microsoft, NextPage, Novell, Statoil, Toshiba, and the United States Library of Congress [1]. This effort resulted in the approval of ECMA-376 in December 2006, which was subsequently fast-tracked into ISO/IEC JTC 1. Ecma submitted the ECMA-376 standard to JTC 1 as DIS 29500 in January 2007 for approval via the fast-track process. During the letter-ballot phase, completed in September 2007, 3,522 comments were received from National Bodies. On 14 January 2008, the Editor, supported by TC45, published a Disposition of Comments document addressing all comments. On 25–29 February, 2008, a Ballot Resolution Meeting took place in Geneva, Switzerland; following this meeting, during one month, the National Bodies had the possibility of changing their vote. The standard met the ISO/IEC DIS approval criteria with 75% of the JTC 1 participating member votes cast positive and 14% of the total of national member body votes cast negative. Publication as ISO/IEC 29500 is expected to occur in October 2008, or soon thereafter. Publication of ECMA-376 Edition 2, which is fully aligned with ISO/IEC 29500, is expected to occur in December 2008.ISO/IEC JTC 1/SC 34 is the committee in charge of maintenance of ISO/IEC 29500, with active participation from Ecma TC45.

7 Publication of the Binary Formats and their Mapping to Office Open XML

The scope of the standard includes neither the Microsoft Office binary formats (.doc, .ppt, and .xls) nor a mapping from those formats to Office Open XML. The interoperability goal has been to ensure that an Office Open XML document written by a conforming producer could be read by a conforming consumer, and that the two would agree on the semantics of that document.

Nevertheless, in connection with the review of the Draft Standard, a number of National Bodies expressed an interest in publication of the binary formats. While these had been available royalty-free under RAND-Z conditions – in fact, members of TC45 from Apple, Novell, and Microsoft were already quite familiar with them – the process for obtaining them was not direct; it included sending a request to Microsoft via email. So, in response to National Bodies' comments, the binary formats were made available for direct download, along with several supporting technologies. Microsoft placed all of these under the Open Specification Promise, and the British Library and the United States Library of Congress agreed to host them so as to guarantee open access in the future. Here are the relevant links:

- Office Binary Formats: http://www.microsoft.com/interop/docs/OfficeBinaryFormats.mspx
- Windows Compound File Format, Windows Metafile Format, and Ink Serialized Format: http://www.microsoft.com/interop/docs/supportingtechnologies.mspx
- Microsoft Open Specification Promise: http://www.microsoft.com/interop/osp/default.mspx
- British Library mirror: http://www.bl.uk/dp/formats
- United States Library of Congress mirror: http://www.digitalpreservation.gov/formats/intro/specifications.shtml

In addition, some National Bodies requested a published mapping from the binary formats to Office Open XML. An open-source effort was started on SourceForge (http://b2xtranslator.sourceforge.net/) to create such a mapping; see the early results at the bottom of the Documentation section of the site. The project is covered by the open-source Berkeley Software Distribution (BSD) license, which allows anyone to use the mapping, submit bugs and feedback, or contribute to the project.

8 Summary

Office Open XML is the product of substantial effort by representatives from many industry and public institutions with diverse backgrounds and organizational interests. It covers the full set of features used in the existing document corpus, as well as the internationalization needs inherent in all of the major language groups worldwide. As a result of the standardization work in Ecma, then in ISO/IEC JTC 1, Office Open XML has enabled a high level of interoperability and platform independence; and its documentation has become both complete (through extensive reference material) and accessible (through non-normative descriptions). It also includes enough information for content creators to include accessible information in their documents and assistive technology products to properly process those documents. Office Open XML implementations can be very small and provide focused functionality, or they can encompass the full feature set. Extensibility mechanisms built into the format guarantee room for innovation.

Standardizing the format specification and maintaining it over time ensure that multiple parties can safely rely on it, confident that further evolution will enjoy the checks and balances afforded by an open standards process. The compelling need exists for an open document-format standard that is capable of preserving the billions of documents that have been created in the pre-existing binary formats, and the billions that continue to be created each year. Technological advances in hardware, networking, and a standards-based software infrastructure make it possible. The explosive diversification in market demand – including significant existing investments in mission critical business systems – makes it essential.

9 CITATIONS

1. **Ecma International.** TC45 - Office Open XML Formats. *Ecma International.* [Online] http://www.ecma-international.org/memento/TC45.htm.
2. **W3C.** XML Schema. *World Wide Web Consortium.* [Online] http://www.w3.org/XML/Schema.
3. **ISO**. ISO/IEC 639 -1,-2,-3: Codes for representation of names of languages. *International Organization for Standardization.* [Online] http://www.iso.org/iso/catalogue_detail?csnumber=22109
4. **ISO.** Information technology – Document Schema Definition Language (DSDL) – Part 2: Regular-grammar-based validation – RELAX NG, ISO/IEC 19757-2:2003. *International Organization for Standardization.* [Online] http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=37605.
5. **ISO.** Information technology – Document Schema Definition Languages (DSDL) – Part 4: Namespace-based Validation Dispatching Language (NVDL), ISO/IEC 19757-4:2006. *International Organization for Standardization.* [Online] http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=38615.
6. **W3C.** Extensible Markup Language (XML) 1.0 (Fourth Edition). *World Wide Web Consortium.* [Online] 2006. http://www.w3.org/TR/2006/REC-xml-20060816/.
7. **W3C.** Namespaces in XML 1.0 (Second Edition). *World Wide Web Consortium.* [Online] 2006. http://www.w3.org/TR/2006/REC-xml-names-20060816/
8. **XBRL International.** XBRL Specifications. *Extensible Business Reporting Language.* [Online] http://www.xbrl.org/Specifications/.
9. **Health Level Seven.** HL7 ANSI-Approved Standards. *Health Level Seven.* [Online] http://www.hl7.org/about/directories.cfm?framepage=/documentcenter/public/faq/ansi_approved.htm.
10. **W3C.** W3C Web Services Architecture. *World Wide Web Consortium.* [Online] 2002. http://www.w3.org/2002/ws/.
11. **W3C.** W3C XML Signature. *World Wide Web Consortium.* [Online] http://www.w3.org/Signature/.
12. **W3C.** W3C XML Encryption. *World Wide Web Consortium.* [Online] 2001. http://www.w3.org/Encryption/2001/.
13. **W3C.** XSL and XSLT. *World Wide Web Consortium.* [Online] http://www.w3.org/Style/XSL/.
14. **W3C.** Hypertext Transfer Protocol – HTTP/1.1. *World Wide Web Consortium.* [Online] http://www.w3.org/Protocols/rfc2616/rfc2616.html.
15. Unicode Bidirectional Algorithm. [Online] http://www.unicode.org/reports/tr9/.
16. RFC4646/BCP 47. [Online] http://www.rfc-editor.org/rfc/bcp/bcp47.txt.
17. PANOSE. [Online] http://www.w3.org/Fonts/Panose/pan2.html.