EXPLANATORY REPORT ON OFFICE OPEN XML STANDARD (ECMA-376) SUBMITTED TO JTC 1 FOR FAST-TRACK

ECMA TC45
TOM NGO (NEXTPAGE), EDITOR

1 INTRODUCTION

Office Open XML (OpenXML) is a proposed open standard 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 the format. Ultimately, all users enjoy the benefits of an XML standard for their documents, including stability, preservation, interoperability, and ongoing evolution.

The work to standardize OpenXML has been carried out by Ecma International via its Technical Committee 45 (TC45), which includes 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 white paper summarizes OpenXML. Read it to:

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

2 PURPOSES FOR THE STANDARD

OpenXML was designed from the start to be capable of faithfully representing the pre-existing corpus of word-processing documents, presentations, and spreadsheets that are encoded in binary formats defined by Microsoft Corporation. The standardization process consisted of mirroring in XML the capabilities required to represent the existing corpus, extending them, providing detailed documentation, and enabling interoperability. At the time of writing, more than 400 million users generate documents in the binary formats, with estimates exceeding 40 billion documents and billions more being created each year.

The original binary formats for these files 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. OpenXML 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 Standard

OpenXML defines formats for word-processing, presentation, and spreadsheet documents. Each type of document is specified through a primary markup language: WordprocessingML, PresentationML, or SpreadsheetML. 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 OpenXML) 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.

Part 1 – Fundamentals 165 pages	–	Defines vocabulary, notational conventions, and abbreviations.

	– Summarizes the three primary markup languages and the supporting markup languages.
	– Establishes conditions for conformance and provides interoperability guidelines.
	– Describes the constraints within the Open Packaging Conventions that apply to each document type.
Part 2 – Open Packaging Conventions 125 pages	– Defines the Open Packaging Conventions (OPC). Every OpenXML 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 issued only in electronic form. The annex also includes non-normative representations of the schemas using RELAX NG (ISO/IEC 19757-2) (3).
Part 3 – Primer 466 pages	– Introduces the features of each markup language, providing context and illustrating elements through examples and diagrams. This Part is informative (non-normative).
	– Describes the facility for storing custom XML data within a package to support integration with business data.
Part 4 – Markup Language Reference 5756 pages	– Defines every element and attribute, the hierarchy of parent/child relationships for elements, and additional semantics as appropriate. This Part is intended for use as a reference whenever complete detail about an element or attribute is required.
	– Defines the facility for storing custom XML data.
	– Declares the XML schemas for the markup languages as XSD (2), in an annex that is issued only in electronic form. The annex also expresses them non-normatively using RELAX NG (ISO/IEC 19757-2) (3).
Part 5 – Markup Compatibility and Extensibility 34 Pages	– Describes facilities for extension of OpenXML documents.
	– Specifies elements and attributes by which applications with different extensions can interoperate.
	– Expresses extensibility rules using NVDL (ISO/IEC 19757-4) (4).

In order to ease reading and navigation through these documents, the electronic versions have many internal active links. In particular, Part 4 has links to parent and child elements throughout.

4 Properties of the Standard

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

- “Interoperability” describes how OpenXML 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 OpenXML supports every major language group.
- “Low Barrier to Developer Adoption”, “Compactness”, and “Modularity” list specific ways in which OpenXML avoids or removes practical impediments to implementation by diverse parties: learning curve, minimum feature set, and performance.
- “High Fidelity Migration” describes how OpenXML 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 OpenXML incorporates business information in custom schemas to enable integration and reuse of information between productivity applications and information systems.
- “Room for Innovation” describes how OpenXML prepares for the future by defining further extensibility mechanisms and providing for interoperability between applications with differing feature sets.

The remainder of this document, including this section, is a topical guide to OpenXML. References to the Specification are all of the form §Part:section.subsection; for example, §1:2.5 refers to Part 1, Section 2.5 of the Specification. References to other headings within this paper are by name.

4.1 INTEROPERABILITY

Developers can write applications that consume and produce OpenXML on multiple platforms.

Foremost, the interoperability of OpenXML has been accomplished through extensive contributions, modification, 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 OpenXML
- Corporations (BP, Barclays Capital, Essilor, Statoil) with heavy investments in existing content, including mission-critical transaction systems
- The British Library and the United States Library of Congress, both of whom have direct interest in preservation

During preparation, committee members raised and resolved hundreds of issues regarding policy, clarity, semantics, and possible dependence on environment. Representative clusters of issues and other 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) and NVDL (ISO/IEC 19757-4) (4)
- Development of tools for automatically analyzing and visualizing the schemas
- Internationalization
- 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 OpenXML 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.

- OpenXML 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 (§1:14.2.12).
- Embedded controls can be of any type, such as Java or ActiveX (§1:15.2.8).
- WordprocessingML font specifications can include font metrics and PANOSE information to assist in finding a substitution font if the original is not available (§3:2.10.5).

In addition, OpenXML 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.
- OpenXML introduces a more general mechanism called the Alternate Content Block (§3:2.18.4), 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”.
- OpenXML 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 OpenXML conforms to open W3C standards such as XML (5) and XML Namespaces (6). This fact alone allows a base level of interoperability across all platforms and operating systems that adhere to these open standards.

4.2 INTERNATIONALIZATION

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

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

Text orientation: OpenXML 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 (§4:2.3.1.6) and the level of a run within a paragraph (§4:2.3.2.28). Similarly, in DrawingML text, it can be controlled on the body level (§4:5.1.5.1.1), on the paragraph level (§4:5.1.5.2.2), and within numbered bullets (§4.5.1.5.4).

Text flow: In WordprocessingML, the direction of text flow can be controlled at the level of a section or a table (§4:2.3.1.41) or at the level of a paragraph (§4:2.3.2.28). At the section and table levels, text flow can be controlled in the vertical and horizontal directions. This allows OpenXML 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 utilize Kumimoji settings at the paragraph and run levels to flow text horizontally and numbers vertically (§4:5.1.5.2.3, §4:5.1.5.3.9). In WordprocessingML (§4:2.3.1.16) and PresentationML (§4:4.3.1.15), character flow can also be specified using Kinsoku settings to specify which characters are allowed to begin and end a line of text.

Number representation: For field formatting in WordprocessingML (§4:2.16.4.3), paragraph/list numbering in WordprocessingML (§4:2.9), and numbering in DrawingML (§4:5.1.5.4, §4:5.1.12.61), numbers can be formatted using any of several dozen number formats, including Hiragana, Arabic, Abjad, Thai, 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 (§4:3.8.30) and in references to external data (§4.3.13.12).

Date representation: In WordprocessingML (§4:2.18.7) and SpreadsheetML (§4:3.18.5), calendar dates can be written using Gregorian (three variants), Hebrew, Hijri, Japanese (Emperor Era), Korean (Tangun Era), Saka, Taiwanese, and Thai formats.

Formulas: The formula specification in SpreadsheetML provides several internationalization-related conversion functions, such as BAHTTEXT (§4:3.17.7.22), JIS (§4:3.17.7.185), and ASC (§4:3.17.7.11).

Language identifiers: In WordprocessingML (§4:2.3.2.18) and DrawingML (§4:5.1.5.3), every paragraph and run can be tagged with a language identifier, allowing an application to select appropriate proofing tools and other language-specific functionality. In addition to an identifier for each language, OpenXML 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

An experienced developer can begin to write simple OpenXML-conformant applications within a few hours of beginning to read the Specification.

Although the Specification describes a large feature set, an OpenXML-conformant 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 the document type [that it expects]” and that a conforming producer “be able to produce conforming documents” (§1:2.5). It also provides Interoperability Guidelines that state the role of element semantics (§1:2.6).

A conformant application can have extremely focused functionality. 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 text 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 OpenXML. 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 OpenXML.
- 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 (§4:2.3.3.13).
- Without implementing formulas and integrating with an external data source, a spreadsheet program could work from cached calculations (§3:3.2.9) and cached external data (§4:3.14 and §4:3.10.1.76).

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

4.4 COMPACTNESS

The OpenXML 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 OpenXML file is conventionally stored in a ZIP archive for purposes of packaging and compression, following the recommended implementation of the Open Packaging Conventions. Perhaps surprisingly, OpenXML 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. One class of examples removes redundant storage of large objects.

- In SpreadsheetML, repeated strings are stored in a string table in the workbook, and referenced by index (§3:3.3).
- 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 (§3:3.2.9.2).

- In DrawingML, shape names (§4:5.1.12.56), text geometries (§4:5.1.12.76), and other presets (several throughout §3:5.8, §3:5.9, and §4:5.1.12) 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 (§4:5.1.11.18).

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

- In WordprocessingML, styles are hierarchical (§3:2.8.9).
- In DrawingML, shapes are grouped hierarchically (§4:5.1.2.1.20).
- In PresentationML, a default hierarchy relates slide masters, slide layouts, and slides (§3:4.2).

Other aspects of OpenXML 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 OpenXML format cooperate to provide this modularity.

- A document is not monolithic; it is built out of multiple parts.
- Relationships between parts are themselves stored in parts.
- The ZIP archive format that is typically used to support OpenXML 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 (§3:13.3.8). 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 OpenXML-specific and therefore do require some knowledge of the relevant markup language (§1:9.2).

4.6 HIGH FIDELITY MIGRATION

OpenXML 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 OpenXML 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, OpenXML 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 OpenXML 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 (§3:2.8, especially §3:2.8.10).
- The PresentationML specification documents the animation features (§3:4.4).

OpenXML 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.

- OpenXML defines effects such as surface appearances (§5.1.12.50) without constraining a developer to match those effects pixel for pixel.
- OpenXML defines parameters such as page margins (§4:2.6.11), font (§4:2.8), and justification (§4:2.3.1.13). It allows developers to implement different flow algorithms as long as they respect those parameters.
- The SpreadsheetML formula specification (§4:3.17.7) 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 (§4:3.17.5).
- 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 (§4:3.17.7.227) 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 (§3:6), are included primarily for backward compatibility. The use of newer standards already in OpenXML, such as DrawingML (§3:5), is encouraged when writing new documents.

4.7 INTEGRATION WITH BUSINESS DATA

OpenXML enables organizations to integrate productivity applications with information systems that manage business processes by enabling the use of custom schemas within OpenXML documents. An organization’s goals in taking this approach would be 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.

OpenXML 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, or use industry standard schemas such as XBRL for financial reporting (7) and HL7 for health-care information(8). 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, as long as it is expressed in XSD form (2).

Second, the custom data are embedded in any OpenXML document in a Custom XML part (§3.7.3) and can be described using a Custom XML Data Properties part (§4:7.5). By separating these custom data from presentation, OpenXML 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

OpenXML is designed to encourage developers to create new applications that were not contemplated when the binary formats were defined, or even when OpenXML 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 OpenXML) 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 OpenXML feature set has two main options:

- Alternate content blocks: An alternate content block (§3:2.18.4 and §5:9.2) 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 (§3:2.6) stores arbitrary custom XML without a visual representation.

Developers have room to innovate outside of those extensibility mechanisms.

- *Alternative interaction paradigms*. OpenXML 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 conformant 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.

- *Novel computing environments*. The Conformance statement admits applications that have low capacity, so that they can run on small devices, and applications that implement only a subset of OpenXML (§1:2.6). The Additional Characteristics mechanism permits a producing application to communicate its capacity limits (§3:8.1).

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 (9). Although such applications have no user-visible behavior other than their operations on data contained within OpenXML documents, they are subject to document conformance (§1:2.4) and application conformance (§1:2.5), which are purely syntactic, and interoperability guidelines (§1:2.6), which incorporate semantics.

While it is impossible to enumerate all possible use cases for customized XML processing, one may anticipate XML-centric services that process OpenXML documents for automatic extraction and insertion of custom data, custom security services such as XML Digital Signature (10) or XML Encryption (11), or even arbitrary XSLT transformations (12) that convert to and from other XML formats. OpenXML places no prohibitions or limitations on such processing.

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 OpenXML 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 OpenXML file by using any ZIP viewer. It is useful to inspect the contents of a small OpenXML 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 OpenXML document is an OPC *package* (§5:8). A package is a flat collection of *parts* (§5:8.1). Each part has a case-insensitive *part name* that consists of a slash-delimited sequence of segment names such as "/pres/slides/slide1.xml" (§5:8.1.1). Each part also has a *content type* (§5:8.1.2). 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 (13).

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 OpenXML document hold OpenXML, 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* (§3:2.1). Each story is one of the following: the main document (§3:2.2), the glossary document (§3:2.13), a subdocument (§3:2.18.2), a header (§3:2.11.1), a footer (§3:2.11.2), a comment (§3:2.14.5), a frame, a text box (§3:2.18.1), a footnote (§3:2.12.1), or an endnote (§3:2.12.2).

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 (§3:2.2):

- `document` – the root element of the main document (§3:2.3).
- `body` – body (§3:2.7.1). Can contain multiple paragraphs. Can also contain section properties specified in a `sectPr` element.
- `p` – paragraph (§3:2.4.1). 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 (§3:2.4.4).
- `r` – run (§3:2.4.2). Can contain multiple types of run content, primarily text ranges. Can also contain run properties (`rPr`). The run is a fundamental concept within OpenXML. 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, OpenXML differs significantly from formats that allow for arbitrary nesting of properties, such as HTML.
- `t` – text range (§3:2.4.3.1). 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 (§3:2.8.10). Those styles are themselves organized into inheritance hierarchies (§3:2.8.9).

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 (§3:4.2), which we list from top to bottom in the default hierarchy:

- slide masters, notes masters, and handout masters (§3:4.2.2), all of which inherit properties from presentation;
- slide layouts (§3:4.2.5), which inherit properties from slide master; and
- slides (§3:4.2.3) and notes pages (§3:4.2.4), 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 (§3:2.2):

- `sld`, `sldLayout`, `sldMaster`, `notes`, `notesMaster`, or `handoutMaster` – the root element.
- `cSld` – slide (§4:4.4.1.15). Can contain DrawingML elements (as described in the next two bullets) and other structural elements (as described below).
- `spTree` – shape tree (§4:4.4.1.42). Can contain group shape properties in a `grpSpPr` element (§4:4.4.1.20) and non-visual group shape properties in an `nvGrpSpPr` element (§4:4.4.1.28). This node and its descendants are all DrawingML elements. We list some DrawingML elements here because of their pivotal role in PresentationML.
- `sp` – shape (§4:4.4.1.40). Can contain shape properties in a `spPr` element (§4:4.4.1.41) and non-visual shape properties in an `nvSpPr` element (§4:4.4.1.31).

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 (§3:3.2); 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 (§3:3.2).
- `sheetData` – the cell table, which represents every non-empty cell in the worksheet (§3:3.2.4).
- `row` – one row of cells in the cell table (§3:2.8).

- `c` – one cell (§3:3.2.9). 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 (§3:3.2.9.1) and optional formula (§3:3.2.9.2) 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 (§3:3.3 and §3:3.2.9.2.1) 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 OpenXML document.

- DrawingML (§3:5) – used to represent shapes and other graphically rendered objects within a document.

- VML (§3:6) – a format for vector graphics that is included for backwards compatibility and will eventually be replaced by DrawingML.

- Shared MLs: Math (§3:7.1), Metadata (§3:7.2), Custom XML (§3:7.3), and Bibliography (§3:7.4).

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 SUMMARY

OpenXML 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 by Ecma TC45 (1) and contributions via public comment, OpenXML 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 assistive technology products to properly process documents. OpenXML 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.

7 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 19757-2:2003. *International Organization for Standardization.* [Online] http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=37605&ICS1=35&ICS2=240&ICS3=30.

4. **ISO.** ISO/IEC 19757-4:2006. *International Organization for Standardization.* [Online] http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=38615&ICS1=35&ICS2=240&ICS3=30.

5. **W3C.** Extensible Markup Language (XML) 1.0 (Fourth Edition). *World Wide Web Consortium.* [Online] 2006. http://www.w3.org/TR/2006/REC-xml-20060816/.

6. **W3C.** Namespaces in XML 1.0 (Second Edition). *World Wide Web Consortium.* [Online] 2006. http://www.w3.org/TR/2006/REC-xml-names-20060816/.

7. **XBRL International.** XBRL Specifications. *Extensible Business Reporting Language.* [Online] http://www.xbrl.org/Specifications/.

8. **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.

9. **W3C.** W3C Web Services Architecture. *World Wide Web Consortium.* [Online] 2002. http://www.w3.org/2002/ws/.

10. **W3C.** W3C XML Signature. *World Wide Web Consortium.* [Online] http://www.w3.org/Signature/.

11. **W3C.** W3C XML Encryption. *World Wide Web Consortium.* [Online] 2001. http://www.w3.org/Encryption/2001/.

12. **W3C.** XSL and XSLT. *World Wide Web Consortium.* [Online] http://www.w3.org/Style/XSL/.

13. **W3C.** Hypertext Transfer Protocol -- HTTP/1.1. *World Wide Web Consortium.* [Online] http://www.w3.org/Protocols/rfc2616/rfc2616.html.