DIS 29500 Technical Comments not Adequately Resolved

Response 0001

This response is inadequate. Correct use of standards control language (shall, shall not, etc.) is not a matter for an un-reviewed "editorial pass". Such changes are at the core of what the normative provisions of the standard are and how conformance is defined, and the definition of conformance is a primary purpose of a standard and critical to procurement decisions. NB's must see, review and approve such changes before deciding whether to approve DIS 29500.

In the NB comments there are listed specific cases of where the control language was unacceptable. It is puzzling that Ecma has decided not to include specific text changes to address these specific points in their proposed Disposition of Comments.

Response 0005

We disagree with the assertion that there "is no confusion" caused by the choice of the name Office Open XML.

We raised concerns[1] last year about the unfortunate confusion caused by the choice of the name "Office Open XML", where mistakes were made by the press, analysts, even in Microsoft press releases.

For example, Microsoft hosts a report on "Document formats and the value of choice in healthcare" on their web site[2]. This report argues against ODF and in favor of "Open Office XML", but repeatedly referring to "Open Office XML", making this error five different times.

Within even the last few weeks we have seen many examples of confusion, by users, by analysts, by the press, and even by Microsoft's own enterprise architects. For example:

- January 11th – Cossacks.org.uk reports "Version 6.1 of Office Mobile finally brings support for **Microsoft's Open Office XML** document formats, over a year after Office 2007 was released to businesses"
- January 14th – WindowsWorldNews.com reports on the Becta report "It also has recommended schools stick with the Open Document Format for storing files, rather than **Microsoft's Open Office XML**"
- January 15th, Pravda, with lead sentence "**Microsoft's Open Office XML** and its competitor OpenDocument Format have always been rivals in the field of document formats."
- January 15th -- eHacks blog announcing availability of a DocX convertor, "Microsoft introduced **a new file format in Microsoft Office 2007 called the Open Office XML** Format or simply known as .docx"
- January 15th – HighPosition.net article on recent EC investigation, "The Commission investigators will see if Microsoft has withheld information needed for interoperability from rival companies. They will also study the **Open Office XML** file format to see if Microsoft is releasing technologies that in effect, reduce compatibility."

1 http://www.robweir.com/blog/2007/01/amusing-but-still-confusing.html

2 http://www.microsoft.com/uk/nhs/content/articles/document-formats-and-the-value-of-choice-in-healthcare.aspx

- January 16th Philippe Destoop, Microsoft's Enterprise Architect for Belgium and Luxembourg, posts a blog entry titled, "Document Standards : **MS Open Office XML** versus ODF".
- January 17th Owen Allen, a Microsoft "technical specialist" reports in his blog the headline "Ecma **Open Office XML** Comments Completed Today"
- January 29th, ComputerWord New Zealand – "A second vote on whether to accept **Microsoft's Open Office XML** document format as an official standard is slated for March."
- February 8th, CNet leads an article with "European Union antitrust officials are looking into whether Microsoft violated regulations in its pursuit of making **Open Office XML** a standard".

This confusion is persistent and widespread. In fact, even the DIS 29500 Project Editor himself seems to have been confused, when in Response 374 he writes: "There is no specific definition of macro languages in the **Open Office XML** specification".

Our recommendation: We reiterate the request that DIS 29500 adopt a different name, one that is not frequently confused with a closely related product.

Response 0016

Although the proposed resolution indeed removes the ST_LangCode, it keeps the legacy language ID table and simply associates it with the "\l" field argument. So, the proposed resolution does nothing but shift some text around and leave the same underlying flaw, namely two ways of identifying languages.

We recommend the sole use of BCP-47 language tags, based on ISO 639. The dual-mode solution offered by Ecma provides no additional expressivity for developers, or any user benefit, but it does increase the cost to implementors.

The need for legacy compatibility is acknowledged, but we believe that BCP-47 tags are directly and unambiguously mappable from the legacy codes, in a lossless way. In fact, at the BRM such a mapping table was provided. It should be the responsibility of those applications and converters that read the legacy binary documents to perform this conversion. It should not be a burden on implementors of OOXML.

Further the resolution of this item was mangled by the BRM, when the Ecma proposed Response was replaced by a NB proposal as BRM Resolution #4. This BRM Resolution missed several of the fixe proposed by Ecma, and has left this language code section in an inconsistent state. For example, the original Ecma Response made changes to more field switches, to /f, /l, /m, etc. The meeting resolution, however, only changes /z.

Response 0018

The changes made at the BRM maintain the legacy date basis as the default. We believe this is bad practice. For most date calculations, especially when dealing with issues of elderly health care or other citizen benefits, serious errors can occur when an application, by default, cannot handle dates before 1900. The default value for dateCompatibility should be false.

Response 0031

Requests for harmonization of OOXML and ODF are seen in the ballot comments of many NB's, including requests from Canada, Korea, South Africa, Peru, France, Belgium and New Zealand.

AFNOR, the French NB, gave a detailed proposal, which said in part:

> After 5 months of extensive discussions between stakeholders in the field of revisable document formats, AFNOR, in the aim to obtain a single standard for XML office document formats within 3 years, makes the following proposal:
>
> - Split the current ECMA 376 standard in 2 parts in order to differentiate the essential OOXML core functions necessary for easy implementation from those functionalities that are needed for the exchange of legacy office file formats;
>
> - Incorporate the technical comments below and those in the attached comment table submitted to the Fast Track;
>
> - Attribute the status of Technical Specification to both parts;
>
> - Establish a process of convergence between ODF (already standardized as ISO/IEC 26300) and the above mentioned OOXML core. ISO/IEC shall invite parties involved to commit themselves to initiate simultaneously the revisions of the existing ODF v1.0 and the OOXML core in order to obtain at the end of the revision process a standard as universal as possible.

New Zealand's proposal was similar:

- OOXML should be considered by JTC 1 for publication as a Type 2 Technical Report.
- Seek to harmonize with the existing ODF standard to reduce the cost of interoperability, cost of having two standards, and cost of support/maintenance .

In addition, the NB's of Great Britain, Brazil, Chile, Columbia, Great Britain, Iran, New Zealand, and the United States requested that specific features be added to OOXML in order to improve interoperability with ISO ODF, in total 40 features such as the ability to have more than 63 columns in a table, to have background images in tables, or to have font weights beyond “normal” and “bold”. These were the exact features that Microsoft's translator project on SourceForge identified as needed to improve translation with ODF[3].

Ecma rejected all of these requests. Ironically, their response was that harmonization is not necessary because there exist tools that will translate between OOXML and ODF. However, those very tools are restricted in their fidelity because of the lack of these features. So their argument is contradictory and circular.

On the question of harmonization, we are either moving toward it, or we are moving away. The Ecma response does not move us toward harmonization, but starts down the road toward further divergence.

---

3 http://odf-converter.sourceforge.net/features.html#hUnsupportedDOCX

Harmonization starts from looking at where the two formats overlap – and there is a significant, perhaps 90%+ area where they do overlap – and expresses the functional overlap in identical markup.

As Tim Bray (co-editor of the W3C XML standard) pointed out in 2005, “The world does not need two ways to say 'This paragraph is in 12-point Arial with 1.2em leading and ragged-right justification'.”

This common functionality between ODF and OOXML would also include a common extensibility mechanism. The remaining 10% of the functionality would represent the focus of the harmonization work. That portion of it which represents a general need could be brought into the core standard. That remaining portion of it which only serves one vendor's needs, such as flags for deprecated legacy formatting options, could be represented using the extensibility mechanism.

Note that translation is a poor substitute for having a single format. If the translation can be done perfectly, then the necessity of two formats is questionable, and if translation introduces errors, it is clearly inferior to having vendors work more to converge their formats.

There is no reason why, by a harmonization process, all of the functionality of Microsoft Office cannot be represented on a base of ISO 26300 OpenDocument Format. In fact, Microsoft acknowledges this much, in a recent statement by Gray Knowlton, Group Product Manager for Microsoft Office, quoted in *PC Word*[4]:

> Also, if individual governments mandate the use of ODF instead of Open XML, Microsoft would adapt, Knowlton said. The company would then implement the missing functionality that ODF doesn't support. However, those extensions would be custom-designed and outside of the standard, which is counter to the idea of an open document standard, Knowlton said.

So we've agreed that this approach is technically feasible. We're also agreed that it is preferable to extend ODF within the standards process, not via custom, non-open extensions. So let's do it! We stand ready and willing to sponsor such a harmonization effort in OASIS, or to participate in a JTC1/SC34 harmonization effort if that is preferred. But let's start harmonization by avoiding further divergence. Introducing a second International Standard for office document formats will only cause confusion.

Response 0035

Ecma has declined to remove these compatibility settings and instead marks them as “deprecated” and moved them into a new annex of the standard (not provided) The words “deprecate” or “deprecated” occur 499 times in Ecma's proposed Disposition of Comments. The BRM, appearing not to like the word “deprecated”,tool these same settings and marked them as “transitory”, a change of name but not of nature.

Ecma claims that these settings, such as “footnoteLayoutLikeWW8” or “lineWrapLikeWord6” are needed for compatibility with legacy documents. However, we note that the legacy documents are not in OOXML format at all. They are in legacy binary, legacy HTML and legacy non-OOXML XML formats. The specification of these proprietary legacy formats exists, and our understanding is that Microsoft is promising to once again make these specifications available freely. Those who desire

4 http://www.pcworld.com/article/id,141424-c,opensource/article.html

100% compatibility with the legacy document formats should obviously avail themselves of the legacy specifications which define or at least describe those formats.

Legacy compatibility comes from applications that understand the legacy formats.

What we have in DIS 29500 are a number of legacy and deprecated settings that don't represent the format of legacy documents, but instead represent how a single vendor's product, Microsoft Office 2007, converts those documents into DIS 29500. The reader should appreciate the circularity of that definition. The legacy features of OOXML are whatever Microsoft Office writes out when converting legacy documents. These features provide no capabilities for any other vendor who wishes to read legacy documents, or even to read OOXML documents which MS Office converted from legacy documents. In fact, the existence of these deprecated features make it more difficult to implement OOXML, not easier. These settings should be removed and expressed, outside of the standard, using application-defined extensions, using the mechanisms of Part 5, Section 11.

Response 0073

The response is insufficient and was not able to be discussed at the BRM.

Formally this “pseudo-code” is meaningless. No programming language has been defined in this standard, nor has one been introduced by external normative reference. Although this pseudo-code appears to be related to C, it contains novelties which are undefined in that that language.

For example what does this mean?

```
if (exists WCTABLE{wcIn})
        return WCTABLE{wcIn}.SecondColumn
```

In C/C++ this is syntactically invalid.

Similarly, we see pseudo-code like:

```
if (exists DBCSTABLE{bytesIn >> 8))
```

This defies analysis even using the pseudo-code's own implied syntax.

There is also a report from one implementor that the provided algorithm is incorrect and does not match the output of legacy versions of Excel.[5]

Response 0075

The Ecma proposed resolution is not acceptable. The codePage attribute remains and allows the specification of ambiguously named, possibly undefined, proprietary code pages. Similar to our concern with Response 0016, we believe that the specification of character set names should exclusively use IANA registered character sets. The burden of mapping these code page names should be performed by the application that initially reads the legacy binary document and writes out

5 http://kohei.us/2008/01/18/excel-sheet-protection-password-hash/

OOXML.

The Ecma proposal given is a step backwards since it altogether removes the table that interprets the legacy character set codes. What does an implementor do now when they see the value “437” for a code page?

Our recommendation: Use IANA character set encoding names exclusively.

Response 0083

The response from Ecma is inaccurate and incomplete.

Ecma says “From the definition of ST_CF it should be clear that this is an open simple type that allows for applications to specify any values they would like. There was a set of enumeration values provided for guidance, but any specific MIME type could also be used (for example, for TIFF; PNG; JPEG)”

However, a look at the XML Schema definition for ST_CF (provided in the electronic annex as vml-spreadsheetDrawing.xsd) shows that this statement is not correct. The definition of ST_CF, shown below, most certainly is an enumeration restriction of xsd:string, limiting the values to EMF, WMF, etc. Any attempt to put the value “PNG” or “JPEG” in a ST_CF element would result in a validation error and would be non-conformant.

So the proposed solution, which changes only the text description of DIS 29500 is inadequate. A corresponding change must also be made to the schema definition in Annex A, which was not provided in Ecma's proposed Disposition of Comments report. Since the original Annex A declares “If discrepancies exist between the electronic version of a schema and its corresponding representation as published in this part, Part 4, the electronic version is the definitive version”, we must believe that our issue is unresolved.

From vml-spreadsheetDrawing.xsd:

```
<xsd:simpleType name="ST_CF">
   <xsd:annotation>
     <xsd:documentation>Clipboard Format Type</xsd:documentation>
   </xsd:annotation>
   <xsd:restriction base="xsd:string">
     <xsd:enumeration value="PictOld">
       <xsd:annotation>
         <xsd:documentation>WMF</xsd:documentation>
       </xsd:annotation>
     </xsd:enumeration>
     <xsd:enumeration value="Pict">
       <xsd:annotation>
         <xsd:documentation>EMF</xsd:documentation>
       </xsd:annotation>
     </xsd:enumeration>
```

```
    <xsd:enumeration value="Bitmap">
      <xsd:annotation>
        <xsd:documentation>Bitmap</xsd:documentation>
      </xsd:annotation>
    </xsd:enumeration>
    <xsd:enumeration value="PictPrint">
      <xsd:annotation>
        <xsd:documentation>Printer Picture</xsd:documentation>
      </xsd:annotation>
    </xsd:enumeration>
    <xsd:enumeration value="PictScreen">
      <xsd:annotation>
        <xsd:documentation>Screen Picture EMF</xsd:documentation>
      </xsd:annotation>
    </xsd:enumeration>
  </xsd:restriction>
</xsd:simpleType>
```

Our recommendation: Update the XML schema to provide the flexibility that the text claims to have. Perhaps rename ST_CF to ST_CF_Legacy and define a new ST_CF which is a union of ST_CF_Legacy and xsd:string?

Response 0084

The BRM did discuss this problem briefly and requested that the behavior be marked “transitional”, but we believe that this resolution is inadequate.

XML has a clearly defined lexical and value space, and the XML strings are interpreted relative to this scheme. Characters outside of this value space have no formal meaning, and we should discourage their use is attributes of type xsd:string. ST_Xstring, bstr and lpstr types are providing an alternative encoding in the lexical space of a string, in order to bring in character values that are otherwise not allowed in XML.

The correct way to handle this is not to use an xsd:string, but to use xsd:base64Binary or xsd:hexBinary or encode control characters using elements like <backspace>.

One XML expert (Rick Jelliffe) has commented[6]:

> ...one area where I think OOXML goes seriously wrong: in a few places it provides a mechanism for circumventing XML’s character repertoire restrictions. I think the idea that just because someone generated an automatic name and used the backspace character as part of it, this should be regarded as acceptable practice in the standard is completely bogus. Several National Bodies have commented on it: I hope ECMA will have the good sense to remove it or severely deprecate it at the least. For example it is clearly a security hole to allow backspace in names, where the visible name may be coded differently than its readers expect: a kind of spoofing.)

---

6 http://www.oreillynet.com/xml/blog/2008/01/the_design_goals_of_xml_1.html

The semantics of strings that contain invalid XML-characters is undefined. For example, OOXML uses ST_Xstring for storing all cell contents in a spreadsheet. But what should an OOXML-supporting application do when given a cell which contains control characters from the C0 control range, forbidden in XML 1.0?

- U+0004 END OF TRANSMISION
- U+0006 ACKNOWLEDGE
- U+0007 BELL
- U+0008 BACKSPACE
- U+0017 SYNCHONRONOUS IDLE

There is a reason XML excludes these dumb terminal control codes. They are neither desired nor necessary in XML.

Elliotte Rusty Harold explains the rationale for this prohibition in his book *Effective XML*:

> The first 32 Unicode characters with code points 0 to 31 are known as the C0 controls. They were originally defined in ASCII to control teletypes and other monospace dumb terminals. Aside from the tab, carriage return, and line feed they have no obvious meaning in text. Since XML is text, it does not include binary characters such as NULL (#x00), BEL (#x07), DC1 (#x11) through DC4 (#x14), and so forth. These noncharacters are historic relics. XML 1.0 does not allow them. This is a good thing. Although dumb terminals and binary-hostile gateways are far less common today than they were twenty years ago, they are still used, and passing these characters through equipment that expects to see plain text can have hasty consequences, including disabling the screen.

Response 0092

The original reported problem was that VML was required in some scenarios, even for new documents. The proposed solution eliminates VML in these scenarios but introduces new XML types, such as:

- CT_ObjectLink
- ST_ObjectDrawAspect
- ST_ObjectUpdateMode
- ST_ObjectLinkType
- ST_ObjectServerFormat
- CT_ObjectPr
- CT_ControlPr
- CT_ObjectAnchor
- CT_CommentPr
- ST_TextHAlign
- ST_TextVAlign

These types are given schema definitions, such as:

```
<complexType name="CT_ObjectPr">
<sequence>
<element name="anchor" type="CT_ObjectAnchor" minOccurs="1" maxOccurs="1"/>
```

```
</sequence>
<attribute name="locked" type="boolean" use="optional" default="true"/>
<attribute name="defaultSize" type="boolean" use="optional" default="true"/>
<attribute name="print" type="boolean" use="optional" default="true"/>
<attribute name="disabled" type="boolean" use="optional" default="false"/>
<attribute name="uiObject" type="boolean" use="optional" default="false"/>
<attribute name="autoFill" type="boolean" use="optional" default="true"/>
<attribute name="autoLine" type="boolean" use="optional" default="true"/>
<attribute name="macro" type="ST_Formula" use="optional"/>
<attribute name="altText" type="ST_XString" use="optional"/>
<attribute name="cf" type="ST_String" use="optional" default="pict"/>
<attribute name="autoPict" type="boolean" use="optional" default="true"/>
<attribute name="dde" type="boolean" use="optional" default="false"/>
<attribute ref="r:id"/>
</complexType>
```

But beyond these XML Schema fragments, there is absolutely no text that explains what these types do or what they mean. So the implementor is left to guess the meaning of attributes such as “justLastX” or “secretEdit” or “dde”. This is unacceptable.

This issue was not discussed at the BRM, so Ecma's flawed response remains.

Response 0093

Instead of adopting an existing open standard, in this case MathML, Ecma claims that such support already exists. But a document created with MathML, following the directions in Ecma's comment, fails to load properly in Office 2007.

What is the problem? Does Office 2007 not support OOXML? Or does OOXML not support MathML?

We would be more comfortable with this claimed support for MathML if Ecma could point us to a single working implementation of this feature.

Response 0099

The original problem was that DIS 29500 contained platform-dependent file paths to refer to images. This would cause problems when reading the same document on multiple platforms, such as Windows and Linux.

Ecma proposes a dual-mode solution, where file paths can be expressed either by OS-dependent path specifications, or by a platform-neutral file-protocol IRI.

This is not acceptable. If even some documents contain OS-dependent paths, then interoperability will suffer. We also note that the legacy behavior maps trivially and flawlessly into the new encoding (using IRI's). This conversion is supported by Microsoft's own Uri .NET class[7] as well as the UrlCreateFromPath() Windows API function[8]. So there remains no good reason to include the legacy behavior at all. It is not needed for compatibility, so it should be eliminated altogether.

7 http://msdn2.microsoft.com/en-us/library/system.uri(VS.71).aspx
8 http://msdn2.microsoft.com/en-us/library/bb773773(VS.85).aspx

Although legacy documents may exist that store paths in Windows-specific formats, the responsibility for normalizing these into portable representations should be with the converting software.

Response 0101

In addressing these comments Ecma adds text that says that the way in which the macro "is located and/or executed" is implementation defined. This statement worries us. Although the runtime API for the macro is reasonably left implementation-defined, the mechanism for locating the macro or command should be explicitly defined for security purposes.

If this is not done then mail gateways, virus scanners, etc., that wish to locate and eliminate such active code elements, or block them, or scan them for virus signatures, will be unable to find them.

Knowing the location of script is also necessary for preserving the integrity of the document when merging or dividing it. If you cannot reliable locate the underlying macro code, then the integrity of the document may broken when doing this.

Since DIS 29500 already has a robust mechanism for specifying parts and their relations, it is suggested that this mechanism be used. For example, an ID could be stored specifying the relationship which contains the macro code.

Our recommendation: Specify fully how to locate the code associated with the macrobutton. This must not be implementation-defined.

Response 102

The Ecma proposal introduces new errors into the standard.

In particular, the "algorithmName" attribute is declared to be an optional xsd:string, and no default value is given. So any value is allowed, or it may be omitted altogether. However, the text does not define what the behavior should be if this attribute is other than those listed in this clause, or if this attribute is omitted.

Also, no mechanism for declaring the use of the deprecated legacy algorithm is given, though it is implied that such should be possible.

Finally, cryptographic algorithms are proposed, such as MD2 and MD4 which are generally considered to be cryptographically weak and therefor "broken". These should not be included in the standard.

Response 110

The semantics of this function is still obscure.

For example, the kpi-property takes on several values which indicate which "KPI component" should be returned.

A value of 1 returns "The actual value, at the time the function is executed".

This is clear enough.

But what exactly does a parameter value of 3 do? It is specified as “The state of the KPI at a specific moment in time”. What is the “state”? Is it the same as the “actual value”? At what “specific moment in time” is intended here? The present? So other time?

Similarly, a kpi-property with a value of 4 indicates “A measure of the value over time”. What does this mean? Is a “measure of the value” the same as the “value”? The same as the “state”? And what does “over time” indicate? That we return some temporal average?

There is not enough information here for anyone to implement this function. This was not discussed at the BRM for lack of time.

Response 0136

The request was to replace PresentationML's animation support in Part 4, Section 4.6 with the existing W3C SMIL (Synchronized Multimedia Integration Language) standard. SMIL, aside from being an already-established standard in this space has the advantage of having already undergone a full, public accessibility review, and was designed to conform with the W3C's Web Accessibility Guidelines[9].

DIS 29500 says of their Animation support:

> This schema is loosely based on the syntax and concepts from the Synchronized Multimedia Integration Language (SMIL), a W3C Recommendation for describing multimedia presentations using XML.

Why only “loosely based” on SMIL? Why not use SMIL directly?

We recommend that animation format be replaced by SMIL. This would also increase interoperability with IS 26300:2006 OpenDocument Format, which also uses SMIL.

Response 0140

Ecma has made no substantive changes to resolve NB comments. Although they crossed out the word “RTF” the clause still allows the use of “application-defined” alternative formats. Implementations are still called on to treat the contents in these alternative formats “as if they were formatted using equivalent WordProcessingML”.

So how exactly does one treat HTML as if it was formatted using “equivalent” WordProcessingML? Since no mapping is provided in the DIS 29500 specification, it is not clear this can be done, if indeed it can be done. For example, HTML, via CSS, allows font weights of 100-900, where 400=normal and 700=bold. So values lighter than normal or heavy than bold are allowed. However, OOXML does not allow anything other than “normal” and “bold”. So it does not appear possible to format HTML in “equivalent” OOXML.

Response 0142

The design of this print settings features is still flawed.

9 http://www.w3.org/TR/SMIL-access/

First, why is the print settings part allowed for SpreadsheetML and WordprocessingML, but not for PresentationML? Ecma claims that this binary blob is necessary for high-fidelity representation of documents. So how are we able to represent presentations without a way of storing their print settings?

Second, not only does this feature, as defined, fail to establish interoperability, but it actively prevents interoperability. The problem is that these printer settings are declared to be "application-defined" but they are all assigned the same type, via the content-type string. So two implementations who wish to exchange compatible printer settings have no way of telling whether their settings are indeed compatible. All print settings have the same type, although they have different, incompatible contents. Given a document, how do I know whether the print settings are in Windows DEVMODE format or MacOS format or Linux format?

Also, Emca says that "A SpreadsheetML package is permitted to contain at most one Printer Settings part per Chartsheet, Dialogsheet,or Worksheetpart". So it is impossible to create a cross-platform document, since only one print settings part can be stored. It can either store Windows print settings or MacIntosh print settings, but not both.

We recommend that this print settings part replaced by a cross-platform XML representation of print settings. In particular we note that Microsoft's own XPS specification defines a PrintTicket markup for which is said:

> Print drivers in earlier versions of Microsoft Windows operating systems used device capabilities functions to communicate printer capabilities to an application and the DEVMODE structure to configure printer settings. The limited extensibility and portability of these methods, however, restrict printer innovation and feature support in applications. For example, custom feature information that appears only in the private portion of the DEVMODE structure is understood only by applications that are aware of that information in advance. The PrintTicket and PrintCapabilities features in Windows Vista, however, support new printer features and innovation by providing a more flexible and extensible framework in which to communicate printer configuration information. Using a common feature schema with the self-describing nature of extensible markup language (XML) makes it much easier for applications to understand new and custom printer features without requiring custom application code.

Isn't this exactly what we want, something in XML without the legacy problems of "limited extensibility and portability"?

This was not discussed at the BRM for lack of time.

Response 0232

The claim by Ecma is that "Office Open XML is compatible with SVG". They then give an example of how SVG can be embedded in an OOXML document. We tried this and the resulting document failed to load properly in Office 2007.

What is the problem? Does Office 2007 not support OOXML? Or does OOXML not support SVG?

We would be more comfortable with this claimed support for SVG if Ecma could point us to a single

working implementation of this new feature.

Response 0341

The given response is incomplete. The “code” attribute has no stated purpose, other than being an “identifier for the current paper size”. So why are we storing an application-dependent code for the paper size (like “240”) rather than the actual label for the paper size, such as “Letter”? This is not portable.

Our request: Define “code” attribute to be the label that describes the paper size. If an application desires to also store an application-dependent numeric code for that paper size, it should use the extensibility mechanisms of Part 5, Section 11 “Application-Defined Extension Elements”.

This was not discussed at the BRM for lack of time.