From: Ted Peters
Sent: Wednesday, February 07, 2001 5:08 PM
To: Bill Gates
Cc: Eric Rudder
Subject: Presentation Format Proposal

Your concern with xmf is that it would be difficult to drive it to critical mass against pdf. I'd like to propose a solution to this problem.

I've spent the last several weeks looking in depth at file formats in the presentation space: flash, pdf, html, xmf, asf, oeb and lit. My conclusion is this: we can define **a unified Microsoft format for presentation**, that covers not only documents but animation, forms, ebooks and media. Rather than take on pdf or flash individually, we can leverage our efforts (ip, tools, evangelism, browser) across media types, justifying the time and capital to drive that format to critical mass

In looking at the various formats, a very clear pattern emerges. The formats are for the most part (minus html) binary, proprietary, and optimized for delivery and consumption (vs. editing). First, there is a header with file information and metainfo. Second, there are indexes for performance (time based, page based, etc.). And third, there are streams of information related to their specific media type, often structured for streaming and often compressed:

- flash - streams of animation data organized by frame (in time)
- pdf - streams of 2d glyphs, strokes and fills, organized by page
- xmf - streams of gdi/gdi+ calls
- asf - generally streams of compressed audio or video
- lit - streams of formatted text

Some of these formats already host multiple streams of different media types. Flash does this (for animation data, audio, script), pdf (for font data, color tables) and clearly, our ASF was expressly designed as a "container" format for multiple arbitrary types. But due to product group silos, ASF has generally been used only for video and audio data. This approach is where we can unify:

We should define and own a proprietary container format for all presentation data...
...AND the stream format for all the major types of media

The container is very much like ASF:

- The header would provide metainfo, document identity, version history, and DRM/security information.
- Optional stream-specific indexes could provide fast access (time, page, etc.). These same stream-specific index anchors can be used for attaching annotations or for providing mail-able references to documents (sending a colleague a specific reference to a particular location in some arbitrary presentation document).
- Streams would provide appropriate formatting and compression for each media type. We would define formats (think codecs) for the important known media types: flowable rich text, 2d data, 3d models, audio, video, image data, animation instruction data, and low-level frozen graphics output. Code (managed code assemblies, for instance) could also be included.
- Multiple streams could be provided for alternative views (different form factors, different quality levels, different client capabilities, internationalization, etc.).

We would provide a universal reader (the browser) that consumes this format, built into our client(s) (for viewing only, not editing). Office might provide an enhanced viewer for collaboration, annotation, group review, etc. Office would also read/write this format for editing. On the server we would provide a back-end for streaming the format (media server - making appropriate decisions on which streams to send given bandwidth) as well as serving up DRM-protected files (a la ebooks' Digital Asset Server).

There are a number of advantages to this approach:

- A universal file format gets all our weight behind one format - this justifies the big push for adoption. Even though the

streams really define the "meat" of the content, this container format will be perceived as a single format
- We build a single file solution for metainfo, drm, document identity -- these are built into the universal container
- We build a single solution for streaming and serving up appropriate streams (from multiply defined ones) based on client context
- Even though there is still stream-specific work to do, we are much more likely to solve the annotation problem (single place for metainfo, "framework" for anchoring)
- We can create different stream formats over time - we can fast-track the existing XMF work on a frozen, print-driver-generated stream without saying that "frozen" is our universal solution. We could immediately support html streams for reflowable/re-editable office docs without strongly promoting html as our universal file format.
- It pushes us towards an single display *engine* that integrates these media types

In many ways, the trident team has been trying to do this unification by pouring all media types into the html (or even xhtml or xml) tree (both file format AND runtime). This approach has been a huge mistake·

- performance for consumption is either not great or prohibitively terrible for certain types (trying to shoe-horn XMF into an tag format killed performance, vml for large 2d sets, size of html+time vs. flash, xml totally irrelevent for audio/video data, storage size of inking data, etc.)
- incrementing from a standards based format leaves us open to rhetorical attack as we innovate
- there's no great way to do packaging (how do you store image data in the xml file, for instance), drm, security, encryption
- there's no easy way to do compression (which is media specific)
- html/xml is not optimized for streamed delivery

With chrisjo's recent org changes (merging duser/trident under mwallent), there's an opportunity for a fresh approach. Defining this new *format* for their new engine to consume might be a galvanizing task. It's something that could be done right away (before starting on *constructing* the new *engine*) that could be evangelized to restore other teams' confidence that the windows presentation platform team was stepping up to solve the problems of content delivery (for ebooks, tablet, dmd, office, flash emulation, etc.)
-T