From: Steven Sinofsky
Sent: Tuesday, March 10, 1998 4:05 PM
To: Bill Gates, Eric Rudder
Cc: Bob Muglia (Exchange), Jon DeVaan; David Cole
Subject: RE: html supporting multiple streams

Message Flag: Follow up
Flag Status: Flagged

This is something we have been working on for the whole product cycle. There are a lot of issues wrapped up in this files/file system complexity issue. I'll try here to talk about them here, but this could easily be the topic of a good memo (we have lots of unimplemented specs and draft proposals on this). Just a thought, but since there are many groups involved it might be a good thing for Eric to bring an outside perspective to, perhaps.

For Office9 we will have our binary 97 file format which will be the preferred way to ship around a single OS file, and HTML+ a folder of supporting files (as henry mentioned below) for viewing in the browser. There are a number of conflicting design issues we need to address across Windows, IE, IIS, as well as Office to make a single file that has all the viewing/browsing behavior we would like. I don't think we can solve this in short order. We are actively working on this--I just don't think we can get everything aligned in one cycle.

This needs to be looked at from the perspective of the viewer/browser/web server and from the perspective of the document author/creation tool.

The reason for the multiple files is, obviously, because that is the way HTML browsers and HTTP servers work today. The entire viewing chain is optimized for multiple streaming OS files. The HTTP server and chache manager are based on this, the browser has a bunch of cache management that uses this, and the end-user browsing experience (the "zippyness") is based on browsers optimizing for multiple GET requests that map to the main HTML page along with additional images (GIF or JPG). All of these streams are read from start to finish and sucked over the wire without any random access or the need to read headers and do seeks. All this means is that HTML+HTTP are currently optimized around the very simple file level GET of files. Even web site authors use this metaphor as a way of making it easy to have only one copy of the corporate logo on a site (in a standard place, everyone just links to).

From the document creation perspective there are two issues: save performance and file mobility. For save performance, the docfile based formats we have in Office 97 are optimal (did I just say docfiles are optimal?). The format each application uses is essentially a memory image (or direct mapping from one) that is only rendered and read by the application code. As you know, each application is a little different--Word is the piecetable, random access format, and Excel has a sequence of BIFF records that all must be read in to open the file, etc. These formats are very poor at being rendered in the HTTP/HTML paradigm described above because they require random access and/or for the entire file to be read before rendering can begin. We have looked at trying to progressively render the binary formats, but this would involve a new binary format to separate out some of the display from the logic.

HTML of course is a format that does not lend itself to incremental rewriting as Word does, so one has to regenerate the whole file at each save (as we do in Excel). We can optimize resaving of graphics, as you saw in the demo, but in general the editor has to output the HTML the way the browser will read it. This is half of why a multipart file (MHTML, multipart html, uses MIME--multipart internet mail extensions-- which just a series of files, usually HTML + images, with a simple header) is slow--you have to rewrite the whole thing at each save. The additional burden of needing to uuencode images is just goofiness because of the goal of making something ASCII for email (actually I don't know if the format really requires ASCII or not). MIME is used by mail and is well suited since most mail messages are only saved a single time. We had in our Office9 plans the idea of using MHTML but the save time turned out to be too slow, since the whole file has to be rewritten each time. There is an even bigger problem in that this format cannot be viewed by any browser (IE5 would have done this, but since we decided not to support they did too--ironically, Communicator supports MHTML in mail but not viewable in Navigator).

File mobility is the biggest end-user problem. First, as a user you just see that a single save throws a bunch of goo into My Documents. This also means that if you want to mail a document as an attachment and include all the subpieces, you can't do this if HTML is your file format. For this scenario you would have to use the Office binary format. On the other hand if you want to mail someone HTML so they can always view it in the browser, then we need to send HTML as browsers universally understand it--that means we need to send a series of HTML+images as separate files. It doesn't make a lot of sense to mail an HTML format that a browser can't read. That is sort of a catch-22 and what makes one lean away from the inferior MHTML solution.

The issue of multiple files is also confusing to people that haven't used other web tools (which means everyone). Having the scenario Henry describes is worrisome.

There are two approaches that are very different. The first one aims to make the editing experience better, whereas the second is all about changing the web model. Both have pros and cons.

(a) Use the shell to essentialy "hide" the notion of multiple files from the user. This can be done by writing a shell extension that understands that an HTML document **FOO.HTM** also has a folder called **FOO FILES/**, but that folder is hidden from the user when the use File Open or the Explorer. The underlying storage is still what the web server and browser expect--a series of plain OS files at the granularity of HTML, GIF, and JPG.

[In Office9 we allow the choice of saving everything to a single flat directory or using the FOO.HTM plus FOO FILES/ approach. In both cases we also emit a "manifest" which is a small XML file that is essentially a directory of the entire file. Note that in Office9 some of hte supporting files might not be HTML/GIF/JPG, but could also be the VBA project or the editing data for an embedded OLE object, such as an Equation. In

IE5/Office9 for example if you do Save As from the browser you will get a set of files that look just like Office created them--you get an HTM, a folder, and the XML directory of files.]

(b) Create a new format that has HTML in it and then solve the end-to-end web distribution of that file. This is more aggressive and requires a lot more infrastructure. In this model, there is an efficient MHTML-like representation (actually it could just be a docfile with an HTML ASCII stream and all the other streams as you would expect). We would then need to have code that runs on the server that knows how to take a given URL (http://myweb/mydoc.htm) and break up the document into the constituent parts. There is also some work to fix up relative links, which is something MHTML handles as well. This is something that needs to happen on the server, but we would also need a universal client "unpackager" so that older IE and other browsers could break the file apart and load it

There are some upsides with (a) which include the fact that it is easier to implement and allows the existing OS file system and web infrastructure to remain 100% up to date. There is a big downside in that there are many file system tools that do not observe shell extensions which could make this design hard to enforce and hard for users to understand when they are in a tool (like Backup, Norton Utilities, Office 97, Win 3.x apps, etc.) that do no observe shell extensions. We have gone back and forth on this solution for Office9 and right now feel that the cost of potentially really confusing people with the illusion of simplicity is higher than the value of faking it. For example, this would not solve mailing an attached around since the document is still HTML+supporting files in a directory.

Alternative (b) is attractive in the long run but requires some really good cooperation across a lot of Microsoft to get this right, along with some industry evangelism to get other web servers, web tools, and browsers to deal with the format. We could do this right by learning from our docfile experience by making useful and easy-to-use APIs (and even code) available to ISVs so they can deal with the format. Of course there are many HTTP tools that also would not understand this format (and would treat it as one unknown blob) analagous to tools in (a) that bypass the shell extensions.

The work would involve at least:

- defining the format, which could be as straight forward as writing out the full "HTML" contents to a series of streams in a docfile. I totally agree with using the docfile implementation for this, rather than come up with a new rewritable stream implementation.
- reading/writing this format in editor applications--this could be a lot of work if we make this a complex new API. It might not be so bad if we just output our existing collection of files to separate streams with a directory.
- Implementing a server side unpackager--this would need to be done so that regular browsers (and web crawlers, site management tools, content indexers, and anyone else who requests things over HTTP) see regular web pages as expected. This would need to integrate with the cache manager, etc. It is important to note that this implies all fetches of these files would involve computation on the server. By unpackaging on the server we also allow for browsers to continue to do their progressive rendering using multiple threads/connections.
- Implement a client side unpackager--this would be used for people that receive one of these packaged files but have a downlevel browser or want to edit or view the file in some less up to date tool.
- Viewing these packages in the browser efficiently, which means either relying on a server to dish out the contents one stream at a time. Note that HTTP does not allow seeking so you would always be stuck reading down the whole file, though the browser could start to display it if we had a convention that the displayable HTML is at the start of the file.

The challenges in defining the format shouldn't be understated. If you take the simple approach of just having the HTML in the docfile, then the format is optimized for easy break up and delivery to the browser. In this case we didn't make any progress on making this format efficient for save/load by the editor. On the other hand if we make the format as efficient as today's binary file for save and load we will make it harder for the unpackaging to take place outside of the context of the creating application. Similarly, the problem of progressively rendering one of these new files is a challenge as well

Finally, we were really sensative to adding a new format for Office9. We knew we were sort of fudging it by adding HTML and at the same time saying the file format was not changing. I think I would worry about the customer experience if there really was a new format that did not have universal viewing--it is the universal (Unix, Navigator, etc.) viewing that allows us to say that we're really not adding a whole new format. If a customer had to choose: binary, HTML, or MS-MHTML I think it could potentially backfire on people. I'm not sure though

There are probably even more considerations than I listed here and the folks in Office, FrontPage, IE5, NT5 that are working on this could add even more. This has been a challenging problem.

So where we are is still trying to figure out which path to take moving forward. For Office9, I am comfortable with telling customers that for local hard drive use or for mailing attached documents, the binary file format is still the way to go. We have Microsoft viewers and third party viewers, and an installed base of Office 97 that will all find this acceptable. To save to a web, to open a document on a web, or to work with documents that are only server based using HTML with multiple files is the best we can do. This is not ideal and doesn't solve Henry's problem below, but it also doesn't hide the problem from people unknowingly.

The current thinking is that optimizing the user's experience with shell extensions along with some special work in mail clients might be the best set of tradeoffs. But this is only where we are now and we keep learning more about how people are reacting to Office9 I'll keep you posted.

-----Original Message-----
From: Bill Gates
Sent: Tuesday, March 10, 1998 11 57 AM
To: Eric Rudder
Cc: Steven Sinofsky; Jon DeVaan; David Cole
Subject: html supporting multiple streams

I disagree with docfile=!html.

You could have a funny URL in the HTML stream that refers to the internal docfile stream

this would leverage all the great docfile work we have done.

It would be easy to do. It is the right way to do it. It doesn't mean we couldn't support some other standard that comes along but it is a solution that helps our customers quickly.

I think this is the only thing that can be done for IE 5/Office 9 with the right performance characteristic.

-----Original Message-----
From: Eric Rudder
Sent: Tuesday, March 10, 1998 11:43 AM
To: Bill Gates
Subject: RE: More Filing Donnybrook?

docfile!=html ...

i think a revised version of MHTML is really what's needed. it would be a good contribution from MS to the industry to do this.

-eric

-----Original Message-----
From: Bill Gates
Sent: Tuesday, March 10, 1998 11:38 AM
To: Steven Sinofsky
Cc: Eric Rudder
Subject: FW: More Filing Donnybrook?

fyi...

-----Original Message-----
From: Bill Gates
Sent: Tuesday, March 10, 1998 11:38 AM
To: Henry Burgess
Subject: RE: More Filing Donnybrook?

Fixing this is my top priority feedback to the product group.

The reason its not easy is that there is no HTML format that does this right - UUencoding the embeds is a bad solution.

I asked Office to work with the OS guys and propose a format that will work for this and support it in IE. We might be able to take advantage of DOCFILE since we put SO MUCH effort into that!! I hope we cna use it for this!

-----Original Message-----
From: Henry Burgess
Sent: Tuesday, March 10, 1998 8:45 AM
To: Bill Gates
Subject: More Filing Donnybrook?
Importance: Low

Hi Bill,

I had a great meeting with Nat Brown from David Vaskevitch's group yesterday. I believe that the right things will happen to storage, it's just going to take longer than I would like. OK, now recalling my communications failure in the old CD-ROM days, I am sending this to you as FYI.

One thing Nat told me is that Office 9 will be storing HTML in a file and then storing any embedded objects as binary in separate files and putting those files in a sub directory. If I understand correctly a Word doc with my bio and picture of me is stored as henry.htm + directory containing henry.bmp. The open and save dialogs for office and IE will hide this directory structure.

I hope it is not something that might have been glossed over in an office presentation or demo. I think this the press, if they understand the implications, could have great fun with this. "When I save my word document it generates all this other stuff, if you are not careful you can get confused or lose part of a document."

I hope you will not circulate this email, as I depend on the product groups good will and I don't want to be seen as that negative guy over in research.

Henry Burgess
MSR
henryb @ microsoft.com

P.S. I like the Slate diary.

From: Steven Sinofsky
Sent: Tuesday, March 10, 1998 4 05 PM
To: Bill Gates; Eric Rudder
Cc: Bob Muglia (Exchange); Jon DeVaan, David Cole
Subject: RE: html supporting multiple streams

This is something we have been working on for the whole product cycle. There are a lot of issues wrapped up in this files/file system complexity issue. I'll try here to talk about them here, but this could easily be the topic of a good memo (we have lots of unimplemented specs and draft proposals on this). Just a thought, but since there are many groups involved it might be a good thing for Eric to bring an outside perspective to, perhaps.

For Office9 we will have our binary 97 file format which will be the preferred way to ship around a single OS file, and HTML+ a folder of supporting files (as henry mentioned below) for viewing in the browser. There are a number of conflicting design issues we need to address across Windows, IE, IIS, as well as Office to make a single file that has all the viewing/browsing behavior we would like. I don't think we can solve this in short order. We are actively working on this--I just don't think we can get everything aligned in one cycle.

This needs to be looked at from the perspective of the viewer/browser/web server and from the perspective of the document author/creation tool.

The reason for the multiple files is, obviously, because that is the way HTML browsers and HTTP servers work today. The entire viewing chain is optimized for multiple streaming OS files. The HTTP server and chache manager are based on this, the browser has a bunch of cache management that uses this, and the end-user browsing experience (the "zippyness") is based on browsers optimizing for multiple GET requests that map to the main HTML page along with additional images (GIF or JPG). All of these streams are read from start to finish and sucked over the wire without any random access or the need to read headers and do seeks. All this means is that HTML+HTTP are currently optimized around the very simple file level GET of files. Even web site authors use this metaphor as a way of making it easy to have only one copy of the corporate logo on a site (in a standard place, everyone just links to).

From the document creation perspective there are two issues: save performance and file mobility. For save performance, the docfile based formats we have in Office 97 are optimal (did I just say docfiles are optimal?). The format each application uses is essentially a memory image (or direct mapping from one) that is only rendered and read by the application code. As you know, each application is a little different--Word is the piecetable, random access format, and Excel has a sequence of BIFF records that all must be read in to open the file, etc. These formats are very poor at being rendered in the HTTP/HTML paradigm described above because they require random access and/or for the entire file to be read before rendering can begin. We have looked at trying to progressively render the binary formats, but this would involve a new binary format to separate out some of the display from the logic.

HTML of course is a format that does not lend itself to incremental rewriting as Word does, so one has to regenerate the whole file at each save (as we do in Excel). We can optimize resaving of graphics, as you saw in the demo, but in general the editor has to output the HTML the way the browser will read it. This is half of why a multipart file (MHTML, multipart html, uses MIME--multipart internet mail extensions-- which just a series of files, usually HTML + images, with a simple header) is slow--you have to rewrite the whole thing at each save. The additional burden of needing to uuencode images is just goofiness because of the goal of making something ASCII for email (actually I don't know if the format really requires ASCII or not). MIME is used by mail and is well suited since most mail messages are only saved a single time. We had in our Office9 plans the idea of using MHTML but the save time turned out to be too slow, since the whole file has to be rewritten each time. There is an even bigger problem in that this format cannot be viewed by any browser (IE5 would have done this, but since we decided not to support they did too--ironically, Communicator supports MHTML in mail but not viewable in Navigator)

File mobility is the biggest end-user problem. First, as a user you just see that a single save throws a bunch of goo into My Documents. This also means that if you want to mail a document as an attachment and include all the subpieces, you can't do this if HTML is your file format. For this scenario you would have to use the Office binary format. On the other hand if you want to mail someone HTML so they can always view it in the browser, then we need to send HTML as browsers universally understand it--that means we need to send a series of HTML+images as separate files. It doesn't make a lot of sense to mail an HTML format that a browser can't read. That is sort of a catch-22 and what makes one lean away from the inferior MHTML solution.

The issue of multiple files is also confusing to people that haven't used other web tools (which means everyone). Having the scenario Henry describes is worrisome.

There are two approaches that are very different. The first one aims to make the editing experience better, whereas the second is all about changing the web model. Both have pros and cons.

(a) Use the shell to essentialy "hide" the notion of multiple files from the user. This can be done by writing a shell extension that understands that an HTML document FOO.HTM also has a folder called FOO FILES/, but that folder is hidden from the user when the use File Open or the Explorer. The underlying storage is still what the web server and browser expect--a series of plain OS files at the granularity of HTML, GIF, and JPG.

[In Office9 we allow the choice of saving everything to a single flat directory or using the FOO.HTM plus FOO FILES/ approach. In both cases we also emit a "manifest" which is a small XML file that is essentially a directory of the entire file. Note that in Office9 some of hte supporting files might not be HTML/GIF/JPG, but could also be the VBA project or the editing data for an embedded OLE object, such as an Equation. In IE5/Office9 for example if you do Save As from the browser you will get a set of files that look just like Office created them--you get an HTM, a folder, and the XML directory of files.]

(b) Create a new format that has HTML in it and then solve the end-to-end web distribution of that file. This is more aggressive and requires a lot more infrastructure. In this model, there is an efficient MHTML-like representation (actually it could just be a docfile with an HTML ASCII stream and all the other streams as you would expect). We would then need to have code that runs on the server that knows how to take a given URL (http://myweb/mydoc.htm) and break up the document into the constituent parts. There is also some work to fix up relative links, which is something MHTML handles as well. This is something that needs to happen on the server. but we would also need a universal client "unpackager" so that older IE and other browsers could break the file apart and load it.

There are some upsides with (a) which include the fact that it is easier to implement and allows the existing OS file system and web infrastructure to remain 100% up to date. There is a big downside in that there are many file system tools that do not observe shell extensions which could make this design hard to enforce and hard for users to understand when they are in a tool (like Backup, Norton Utilities, Office 97, Win 3.x apps, etc.) that do no observe shell extensions. We have gone back and forth on this solution for Office9 and right now feel that the cost of potentially really confusing people with the illusion of simplicity is higher than the value of faking it For example, this would not solve mailing an attached around since the document is still HTML+supporting files in a directory.

Alternative (b) is attractive in the long run but requires some really good cooperation across a lot of Microsoft to get this right, along with some industry evangelism to get other web servers, web tools, and browsers to deal with the format. We could do this right by learning from our docfile experience by making useful and easy-to-use APIs (and even code) available to ISVs so they can deal with the format. Of course there are many HTTP tools that also would not understand this format (and would treat it as one unknown blob) analagous to tools in (a) that bypass the shell extensions.

The work would involve at least:

* defining the format, which could be as straight forward as writing out the full "HTML" contents to a series of streams in a docfile. I totally agree with using the docfile implementation for this, rather than come up with a new rewritable stream implementation.
* reading/writing this format in editor applications--this could be a lot of work if we make this a complex new API. It might not be so bad if we just output our existing collection of files to separate streams with a directory.
* Implementing a server side unpackager--this would need to be done so that regular browsers (and web crawlers, site management tools, content indexers, and anyone else who requests things over HTTP) see regular web pages as expected. This would need to integrate with the cache manager, etc. It is important to note that this implies all fetches of these files would involve computation on the server By unpackaging on the server we also allow for browsers to continue to do their progressive rendering using multiple threads/connections.
* Implement a client side unpackager--this would be used for people that receive one of these packaged files but have a downlevel browser or want to edit or view the file in some less up to date tool.
* Viewing these packages in the browser efficiently, which means either relying on a server to dish out the contents one stream at a time. Note that HTTP does not allow seeking so you would always be stuck reading down the whole file, though the browser could start to display it if we had a convention that the displayable HTML is at the start of the file.

The challenges in defining the format shouldn't be understated. If you take the simple approach of just having the HTML in the docfile, then the format is optimized for easy break up and delivery to the browser. In this case we didn't make any progress on making this format efficient for save/load by the editor. On the other hand if we make the format as efficient as today's binary file for save and load we will make it harder for the unpackaging to take place outside of the context of the creating application. Similarly, the problem of progressively rendering one of these new files is a challenge as well.

Finally, we were really sensative to adding a new format for Office9. We knew we were sort of fudging it by adding HTML and at the same time saying the file format was not changing. I think I would worry about the customer experience if there really was a new format that did not have universal viewing--it is the universal (Unix, Navigator, etc.) viewing that allows us to say that we're really not adding a whole new format. If a customer had to choose: binary, HTML, or MS-MHTML I think it could potentially backfire on people. I'm not sure though.

There are probably even more considerations than I listed here and the folks in Office, FrontPage, IE5, NT5 that are working on this could add even more. This has been a challenging problem.

So where we are is still trying to figure out which path to take moving forward. For Office9, I am comfortable with telling customers that for local hard drive use or for mailing attached documents, the binary file format is still the way to go. We have Microsoft viewers and third party viewers, and an installed base of Office 97 that will all find this acceptable. To save to a web, to open a document on a web, or to work with documents that are only server based using HTML with multiple files is the best we can do. This is not ideal and doesn't solve Henry's problem below, but it also doesn't hide the problem from people unknowingly.

The current thinking is that optimizing the user's experience with shell extensions along with some special work in mail clients might be the best set of tradeoffs. But this is only where we are now and we keep learning more about how people are reacting to Office9. I'll keep you posted

-----Original Message-----
From: Bill Gates
Sent: Tuesday, March 10, 1998 11:57 AM
To: Eric Rudder
Cc: Steven Sinofsky, Jon DeVaan, David Cole
Subject: html supporting multiple streams

I disagree with docfile=!html

You could have a funny URL in the HTML stream that refers to the internal docfile stream.

this would leverage all the great docfile work we have done.

It would be easy to do. It is the right way to do it. It doesn't mean we couldn't support some other standard that comes along but it is a solution that helps our customers quickly.

I think this is the only thing that can be done for IE 5/Office 9 with the right performance characteristic.

-----Original Message-----
From: Eric Rudder
Sent: Tuesday, March 10, 1998 11:43 AM
To: Bill Gates
Subject: RE: More Filing Donnybrook?

docfile!=html ...

i think a revised version of MHTML is really what's needed. it would be a good contribution from MS to the industry to do this.

-eric

-----Original Message-----
From: Bill Gates
Sent: Tuesday, March 10, 1998 11:38 AM
To: Steven Sinofsky
Cc: Eric Rudder
Subject: FW More Filing Donnybrook?

fyi...

-----Original Message-----
From: Bill Gates
Sent: Tuesday, March 10, 1998 11:38 AM
To: Henry Burgess
Subject: RE: More Filing Donnybrook?

Fixing this is my top priority feedback to the product group.

The reason its not easy is that there is no HTML format that does this right - UUencoding the embeds is a bad solution.

I asked Office to work with the OS guys and propose a format that will work for this and support it in IE. We might be able to take advantage of DOCFILE since we put SO MUCH effort into that!! I hope we cna use it for this!

-----Original Message-----
From: Henry Burgess
Sent: Tuesday, March 10, 1998 8:45 AM

To: Bill Gates
Subject: More Filing Donnybrook?
Importance: Low

Hi Bill,

I had a great meeting with Nat Brown from David Vaskevitch's group yesterday. I believe that the right things will happen to storage, it's just going to take longer than I would like. OK, now recalling my communications failure in the old CD-ROM days, I am sending this to you as FYI.

One thing Nat told me is that Office 9 will be storing HTML in a file and then storing any embedded objects as binary in separate files and putting those files in a sub directory. If I understand correctly a Word doc with my bio and picture of me is stored as henry.htm + directory containing henry.bmp. The open and save dialogs for office and IE will hide this directory structure.

I hope it is not something that might have been glossed over in an office presentation or demo. I think this the press, if they understand the implications, could have great fun with this. "When I save my word document it generates all this other stuff, if you are not careful you can get confused or lose part of a document."

I hope you will not circulate this email, as I depend on the product groups good will and I don't want to be seen as that negative guy over in research.

Henry Burgess
MSR
henryb @ microsoft.com

P.S. I like the Slate diary.