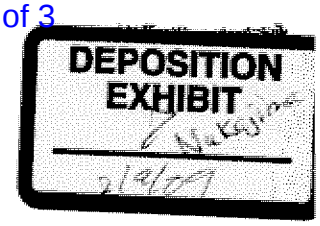
PX0176



From: Satoshi Nakajima
To: Brad Chase; Brad Silverberg; David Cole; Joe Belfiore
Cc: Chris Guzak; Dave Seres; Kurt Eckhardt; Satoshi Nakajima
Subject: RE: Chicago/OLE Q&A
Date: Thursday, May 12, 1994 8:38AM

I think following messages are pretty confusing. We should say that the Chicago shell uses OLE's programming model. Even though we don't actually call OLE32.DLL to load extension DLLs, the shell uses the exactly same algorithm when it loads shell extensions (the shell has a sub-set of OLE's COMPOBJ feature in it -- light weight binder) as OLE does. We took this approach because of two reasons (4Mb goal, and development cycle) and we should not confuse ISVs with it. This is very important to tell ISVs that their shell extension DLLs are OLE's In-Proc Servers, and those shell extension DLLs are OLE-compatible not only in future versions of windows, but also in THIS version. Those DLLs can be loaded by either the shell or OLE today and tomorrow. This compatibility is the key of this technology, and we should empasize it.

I put my specific comments with ###.

-Satoshi

From: Joe Belfiore
To: bradc; bradsi; davidcol
Cc: chrisg; kurte; satona
Subject: RE: Chicago/OLE Q&A
Date: Thursday, May 12, 1994 2:06AM

Does this sound bogus to you, or am I just not really understanding what we are likely to do?

Does Chicago use OLE D/D?
If you can drag paragraph from word into a folder to create a shortcut to the paragraph, that's pretty much using OLE D/D, isn't it?

Q: If the Chicago shell doesn't use OLE, that means that it can't do drag-drop with OLE2 applications, right?
...not if we support the scenario above. Aren't we moving fwd with this delayed initialization of OLE when we get the drop message from OLE? we'd be using OLE d/d then, wouldn't we? (or would we not really get the same drop message that we'd have gotten as a regular OLE client?)

From: Dave Seres
To: Brad Chase; Brad Silverberg; David Cole; Joe Belfiore; Paul Maritz; Richard Barton; Richard Freedman
Cc: Cameron Myhrvold; Developer Relations Staff; Objects/Core; Richard Tong; Roger Heinen; Shauna Braun; WOSA API Marketing
Subject: Chicago/OLE Q&A
Date: Wednesday, May 11, 1994 7:25PM
Priority: High

Here's our latest Chicago/OLE Q&A. Please forward as appropriate.
--Dave

Q: Does Chicago take advantage of OLE 2?
A: Absolutely. Chicago is a great platform for OLE 2 applications.

MSC 00514442

HIGHLY CONFIDENTIAL

MX3171070
CONFIDENTIAL

Chicago incorporates the same high performance 32-bit version of OLE that ships with Daytona. With this new version of OLE, 32-bit applications are fully interoperable with 16-bit applications. In addition, ISVs can exploit OLE's new multi-threading capabilities allowing an application to do several operations concurrently. Chicago's mail system itself takes advantage of these new features. The mail system and the new WordPad and Paint applets support OLE Visual Editing and OLE Drag and Drop. The overall Chicago user interface is modeled on OLE making it very consistent and easy-to-learn.

Q: Does the Chicago shell use OLE 2? What about the Explorer?
A: No. The new 32-bit implementation of OLE2 was not available early enough in the Chicago development cycle to make it a practical candidate for supporting the Chicago shell extensions. This does not affect users. Future versions of the shell will support OLE. The same is true for the Explorer.
The answer must be Yes. To achieve our size goal, we decided to put a sub-set implementation of OLE 2 (light-weight binder) in the shell (so that we can run the shell and old Windows apps without loading OLE2), but it uses the same algorithm when loading In-Proc server DLLs. When we switch to the real OLE2, nobody will notice the difference.

Q: Are context menus and property sheets based on OLE 2?
A. No. These are all part of the Chicago shell. However, they do follow the OLE 2 user interface model so they are consistent from the user perspective. Only a limited number of programmers writing shell extensions will notice the difference.
They follow the OLE2 programming model (In-Proc server) as well. The shell just defines some new additional "interfaces".

Q: Does Chicago support OLE 2 Drag-drop? What about the shell?
A: Yes, Chicago supports the same OLE 2 Drag-drop capabilities now available with Window 3.1. However, the shell does not because 32-bit OLE was not available in time. In the future, it will.
The Chicago shell uses Win 3.1-compatible drag-drop code when the user start the dragging from the shell, therefore, all the existing Win 3.1 application programs will receive same messages (WM_DROPFILES) as they did from the Win 3.1 file namager. All the new OLE2 application programs should continue to support WM_DROPFILES messages as well as being a OLE drop target. On the other hand, when the user start the dragging from OLE 2 apps, the Chicago shell will be one of drop targets -- so that the
user can drop OLE objects and/or links to them on the desktop or any file system folders. The latter feature (drag-drop from OLE apps) is not available
at the beta release yet, and we may cut it from the first release.

Q: If the Chicago shell doesn't use OLE, that means that it can't do drag-drop with OLE2 applications, right?
A: Drag-drop is the one piece of OLE-based interoperability that is important to end-users. Our current plan is to get OLE2 drag-drop working before Chicago is released so that the shell can interoperate at the highest level with OLE2 applications. But even without that, there will be Windows File Manager drag-drop interoperability, which is quite useful.
Read comments above. This is VERY important to tell ISVs that they should continue to support WM_DROPFILES messages, if they want to be a drop target of files.

Q: What was Chicago worried about that it didn't use OLE 2 in its shell? I thought Microsoft was committed to OLE?
A: The fact that the Chicago shell did not employ OLE was simply a matter of scheduling and resource availability. Future versions of the shell will support OLE. Microsoft is firmly committted to OLE. All

major Microsoft applications are currently supporting OLE.

Q: I've heard the shell uses something called "OLE-lite". What is that?
A: There is nothing called "OLE-lite". The shell does not use OLE.
Please don't say such a lie. ISVs will notice that the Chicago shell is able to load In-Proc server DLLs without loading OLE32. We should honestly mention that the Chicago shell has a sub-set implementation of OLE, which enable the shell to load In-Proc server DLLs.

Q: Why does the shell extension model look exactly like OLE? It uses the registry database, GUIDs, has things called "in-proc handlers," and even uses identical interfaces to those found in OLE.
A: The Chicago shell team was clearly influenced by the OLE design, and they re-used OLE concepts and didn't invent entirely new concepts or new pieces of code, like a new Windows registry. However, due scheduling problems -- basically, the lack of 32-bit OLE -- they had to proceed down their own path. We plan to reconverge these technologies in the future.
This message is wrong again. We followed the OLE's programming model so that we can switch to the real implementation without asking ISVs to re-write their extensions. This is very important information to be sent to ISVs. Even if the shell started using the implementation in OLE32.DLL, the Chicago shell extensions will continue to work.

Q: Isn't Chicago shipping with an incomplete version of OLE 2?
A: No. As a matter of fact, it is shipping with the most advanced version of OLE 2 available with support for 32-bit applications, 16/32-bit interoperability and multithreading.

Q: Are today's OLE 2 applications fully upward compatible with OLE 2?
A: Absolutely. 16-bit applications using OLE 2 run great on Chicago. You can also mix 16- and 32-bit applications. You can have 16-bit containers and 32-bit object servers and vice-versa.

Q: Will OLE 2 applications run faster on Chicago?
A: Its too early to be giving definitive results. Both the Chicago and OLE teams are still tuning their code. We can say that OLE was designed to be 32-bit from the beginning so the OLE libraries are naturally more efficient on Chicago. In addition, applications written to exploit multi-threading are perceived by the user to run faster because several applications can run concurrently.

Q: Does Chicago support distributed OLE?
A: No, but future versions of Windows will support distributed objects. Cairo will be the first Windows platform to offer distributed object support with OLE. Follow-on releases to Chicago will also offer OLE distributed object support.

Q: How will Chicago and Cairo run together?
A: Chicago will be a great client to Cairo servers, and will be enhanced after the first release to support new features found in Cairo, such as its distributed file system.

MSC 00514444

HIGHLY CONFIDENTIAL

MX3171072
CONFIDENTIAL