EXHIBIT
Ludwig - 4
1/21/09 - JA

PX0531

Novell

MICROSOFT MEMO -- DRAFT

Date: 10/13/93

To: Paul Maritz, Jim Allchin, Brad Silverberg

From: John Ludwig

Subject: Novell's AppWare

Novell's AppWare has gotten a tremendous amount of press coverage recently, and analysts are increasingly enamored of it. This memo outlines the goals of AppWare, the AppWare architecture, and Novell's major claims for AppWare. Agains each of these I outline possible MS responses. Attached for reference are the AppWare materials provided by Novell at the October Brainshare event.

* * *

AppWare Goals

Novell's Brainshare presentations define the Appware goals as:

> *"To stimulate growth of network applications by*
>
> - *Hiding complexity of the network*
> - *Increasing development efficiency by standardizing access to services"*

An oft-repeated goal in the materials is to provide a cross-platform (client platform that is) solution, reflecting the increasing heterogeneity of customer networks.

⇒ MS response:

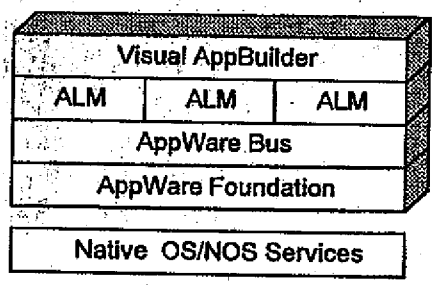
- These are fine goals, we support them.
- WOSA is our framework for standardizing access to network services. WOSA provides single, consistent API access to multiple backend services on Unix, Netware, NT, and hosts.
- We've been delivering on WOSA for several years -- MAPI, ODBC, Windows Sockets are all shipping. WOSA is not vaporware.
- We're layering our OLE2 object model on top of WOSA to make service access even easier, we've been at this for several years and have broad industry support for the OLE model.
- We are making OLE2 and the Windows API available cross-platform to enable cross-platform app deployment while preserving investment in existing Windows applications and tools. AppWare requires rewriting of all this code.

MS 0115589
CONFIDENTIAL

- There is a huge array of tools available already for the Windows API and OLE development, targetting all levels of developers. There is 1 tool announced for AppWare.

AppWare Architecture

AppWare consists of 4 major components: the AppWare Foundation, the AppWare Bus, ALMs or AppWare Loadable Modules, and the Visual AppBuilder.



The **AppWare Foundation** provides a "common, cross-platform set of APIs...(which) allows developers to maintain a single-source base for all development platforms." Basically, this layer virtualizes all services of the underlying OSes on which it is hosted, insulating the developer from differences in these platforms. The services that are provided by this layer include:

Foundation Series	Connectivity Series	User Interface Series
Character Handling	Event Management	App Instance support
Data conversion and storage	Pipe and Socket management	Box Instance support
Error Handling	Object linking	Button Instance support
File IO and file system management	Clipboard	Dialog control
Font Control	Task Manipulation	Display Instance support
Graphics Package		Edit Text Instance support
Instance Management		Help Instance support
Keyboard Management		Interface data management
Memory Management		List Instance support
Message Passing		Menu Instance support
Module Management		Slider Instance support
Pointer/cursor manipulation		Common Dialog support
User preference management		Table Instance support
Printing support		Void Instance support
Resource Management		Window Instance support
OS attributes		

The upper interfaces of the AppWare Foundation are intended to be used by developers developing in C and C++, including the developers of the AppWare Bus from Novell. The Foundation interfaces will be documented in an SDK to be released shortly.

⇒ MS Response:

- The AppWare Foundation is an entirely new OS API. It offers virtually all the services of the OSes it is hosted upon, but with a brand new and different API set. All code has to be rewritten to use it, it provides no migration path from existing code bases.
- The AppWare Foundation is no simpler than any other OS API. The learning curve for it will be just as steep as that for Windows or the Mac or any other API set. Developers are already down these curves.
- Mixed mode development is not possible -- you can't easily write an app that is 1/2 native Windows, 1/2 AppWare, as the systems will not coordinate access to the screen or other resources. You have to commit to the AppWare environment fully.
- The Foundation, as a layer on the host OS, will always be slower and fatter than native code. You can't add a layer without adding size and losing speed.
- The Foundation is incomplete. It does not represent all the services of the host OSes. A single small development team cannot maintain pace with the thousands of developers at MS, Apple, Taligent, IBM, Sun, who are constantly innovating on these platforms.
- The Foundation is behind. By the time it ships, the next revisions of Windows and other OSes will be on the market, requiring a new version of the Foundation.
- The Foundation lacks some critical services, such as threads. This makes the Foundation basically incapable of taking advantage of the multiprocessing capabilities of the underlying OSes.

The layer above the AppWare Foundation is the **AppWare Bus**. The AppWare Bus provides "standardized access to prebuilt application services". The exact services provided by the AppWare Bus are not well documented, but conceptually the Bus seems to be the object model and object layer. The Bus handles bindings and interactions between objects. Per Novell, the object model will be compatible with OLE2.

⇒ MS Response

- The AppWare Bus is an entirely new object model. It is not OLE, it is not CORBA, it is not OPENDOC, it is a new thing. Everything has to be rewritten to it, there is no migration path from existing code bases.

- This new object model will sit side-by-side with the existing native object models of the various platforms -- more code, slower and fatter systems.
- Interoperability with existing native object models, and the hundreds of applications written to them, will never be perfect as the native object models evolve and the AppWare Bus must constantly be revised to keep up.

Residing on the AppWare Bus are **ALMs**, or **AppWare Loadable Modules**. These are objects, provided by Novell and others. The objects that Novell says they will provide (or will be provided by 3rd parties) include:

- File, Print
- Directory Services
- Messaging
- Database
- OLE/DDE
- Software Dist/Licensing
- Telephony
- Image/Document Management
- Network Management
- Multimedia
- Comm
- Security

⇒ MS Response:

- Again, these ALMs have to be completely rewritten from current code. There is no migration path from current drivers/libraries providing these features.
- The list of ALMs to be provided is vaporware. Windows and other platforms have these services now.
- The ALMs presented at Brainshare are not multi-vendor, heterogeneous ALMs. The directory ALM supports NDS. The Telephony ALM supports Netware Telephony services. The File/Print ALMs support Netware. Novell has promised to support other nets only as "market demand warrants". Other nets are supported today by WOSA.
- The ALM SDK is not shipping until the end of year at best. OLE SDKs are available now.

Finally, residing on top of the AppWare framework is **Visual AppBuilder**, a visual programming tool targetted at corporate developers. Constructing an application in Visual AppBuilder is a very graphical process -- lots of dragging and dropping of objects, dragging to establish relationships/actions between

objects, etc. There is little/no direct editting of code involved in creating an application.

Visual Appbuilder is not the only tool that Novell hopes customers use to develop AppWare apps. The Foundation, Bus, and ALM APIs are all published and Novell is encouraging ISVs to provide a variety of tools targetted at different levels of development expertise.

⇒ MS Response:

- Visual AppBuilder is a fine tool. It is too bad it can't be used with the native OSes on which it is hosted.
- Visual AppBuilder is one tool, targetted at one class of developer. There are hundreds of tools available on Windows and other platforms targetted at all levels of developers.
- Building substantial applications in Visual AppBuilder will be quite challenging, as it lacks a traditional IDE coding environment. All "code" must be entered as drag/drop actions and mouse clicks, there is no heavy duty coding environment.

AppWare Claims

Novell claims that the sum of the AppWare elements leads to greater developer productivity due to several characteristics of the environment.

- Application Performance. Apps based on the AppWare Foundation provide the same level of performance as applications based on native implementations.

  ⇒ MS Response: This is nonsense. The foundation is layered on the native OS services and is no simpler by Novell's admission; the layer adds code which means greater size and lower performance.

- Toolkit functionality. The AppWare Foundation provides developers all the functionality they need -- GUI, OS services, inter-app connectivity.

  ⇒ MS Response: Again this is nonsense. Critical services like threads are missing, and are unlikely to be supported on platforms such as the Mac. The AppWare team due to its small size and separation from OS vendor teams will be consistently behind the capabilities of the native OSes. AppWare is not a superset, it is a least common denominator approach.

- Toolkit modularity. The AppWare Foundation architecture can be scaled, based upon the nedds of each application.

  ⇒ MS Response: This is true of each native OS, you can just use the capabilities you need. And since you cannot effectively do mixed-mode development, you can't use just part of AppWare -- you need to make a wholesale conversion with the attendant learning curve.

- Toolkit extensibility. The AppWare Foundation tookit enables developers to extend functionality by implementing featuers not directly provided by the toolkit.

  ⇒ MS Response: But you told us 2 points ago that the AppWare Foundation was complete and never needed to be extended. Which is it? And if I do extend it, the primary benefit claimed of portability is gone as I've now added platform-specific features.

- Development migration path. The AppWare Foundation architecture provides a method of mixing new and legacy code, since most organizations cannot afford to migrate an entire app to a new platform all at once.

  ⇒ MS Response: There is really no mixing supported. You can't draw on the screen using AppWare Foundation calls and native Windows calls and achieve any sensible and manageable results. Converting to AppWare is converting to a brand new OS API, and you need to do it completely.

- Integration with other tools. The architecture allows developers to work with their favorite 3rd party tools and easily integrate their own tools.

  ⇒ MS Response: So where are the tools? There are hundreds of compilers, IDEs, interpreters, debuggers, code management tools, etc. available for Windows and other OSes. So far, AppWare has one committed tool -- Visual AppBuilder -- and one promise from Borland to support it in the future.

- Portability. Applications developed to the AppWare Foundation or to ALMs/AppWare Bus will be portable to any of the AppWare Foundation-supported OSes.

  ⇒ MS Response: Yes, if you use only the services of the AppWare Foundation, and the provided ALMs which work only with Netware servers, then your app will be portable to other platforms supported by the AppWare Foundation.

  But most developers will want applications to be portable across server backends -- different SQL databases, different mail backends. This will require services outside of the AppWare environment. And with the increasing popularity of Windows API support across all the popular client OS platforms (OS/2, Taligent, Unix, etc), writing to Windows is a more compatible, lower learning curve approach to cross-platform development.