Novell AppWare

A System for Developing Network Applications

White Paper

July 1993

For more information about Novell products,
contact Novell as follows:

In the U.S. or Canada: Call 1-800-NETWARE (1-800-638-9273).
In all other locations, contact your local Novell office or call 1-801-429-5588.

Using This Document

This paper provides an overview of AppWare, Novell's system for developing network applications. It describes the components of the AppWare system and explains how developers and users will interact with this new layer of software.

The AppWare system is comprised of two major software components: the AppWare Foundation and the AppWare Bus. These components provide a consistent set of platform-independent, network-independent and service-focused interfaces and engines that accommodate the needs of commercial and corporate developers who need to create network applications quickly and easily.

This paper is divided into the following sections.

AppWare: An Overview gives an overview of the development challenges AppWare addresses.

AppWare: A Technical Description discusses the two major components of AppWare, as well as the wide range of third-party tools and support services for AppWare.

Appendices cover the technologies discussed in this paper in more detail.

Glossary describes the key terms introduced in this paper.

AppWare: An Overview

What Is AppWare?

AppWare is a new layer of software that leverages today's popular operating systems, development tools and applications. AppWare allows both commercial and corporate software developers to create and deploy network applications more quickly and easily. The role of the AppWare layer is similar to the role of the operating system layer. By shielding developers from the complexities of hardware, the operating system has accelerated the growth of new desktop applications. Similarly, AppWare will shield developers from the complexities of networks and accelerate the growth of network applications. As a result, users can more easily run a wide variety of applications that take full advantage of the network and its powerful services.

The Need for AppWare

AppWare provides a solution to two critical challenges currently facing the software development community:

- It simplifies the development of network applications in a complex, heterogeneous environment.

- It shortens the network application development process and makes it more efficient.

Developers who create network applications must deal with the growing numbers of operating systems, development application programming interfaces (APIs), computing standards and development toolkits available today. AppWare hides these complexities from developers by providing them one uniform set of APIs for accessing different operating systems, graphical user interfaces (GUIs) and network services. Using AppWare, programmers can write the application code once and run the application on different operating systems and networks.

Even with a single API and one code path, however, corporate and vertical software developers cannot afford to create applications using line-by-line coding -- a time-consuming task that requires specialized skills. AppWare gives programmers the ability to build applications by using large-grained, interchangeable software components. This way, developers can quickly construct powerful, reliable applications without writing a single line of code.

With AppWare, developers can quickly and easily build applications that provide users the full benefits of the power of their networks. As the leading provider of network operating systems and services, Novell has developed AppWare to provide the underlying APIs, development technologies and services required to successfully write network applications.

Development Challenges

The Complexity of Developing Network Applications in a Heterogeneous Environment

On average, one major new operating system or service API is delivered to developers every 45 days. This rapid pace makes it virtually impossible for application developers to keep up with emerging technologies. Resource constraints force developers to choose among the available alternatives. Often, developers can only afford to focus on one platform or operating system, limiting the markets and minimizing the users' needs that their products can address. Consider this challenge in the following contexts:

Multiple Networking Models

Application developers are currently faced with an unprecedented need for network access and functionality. Now that networks dominate the

computing environment, software developers are shifting their focus from writing standalone applications to creating network applications. Developers must contend with a variety of network models to deliver the services and data their users require.

The widespread presence of heterogeneous computing environments has made the choice of development platforms difficult, as developers try to anticipate the best markets for their products. In the near future, network accessibility and functionality will determine the success or failure of many organizations.

Multiple Desktop Platforms

To support multiple platforms, developers must learn all the details of the desktop platforms on which their applications run. For example, if a developer had to write a million lines of C code to create an application for MS Windows, that developer would likely have to rewrite most of that code for each platform on which the application will run, such as Macintosh, UNIX or OS/2. Furthermore, developers must know how to navigate and control the networks that link these various desktop platforms. Thus, in most cases, the knowledge, skills and resources required to create network applications have been prohibitive.

The Inefficiency of Developing Network Applications

Assume that the complexity problem was solved and programmers had access to all major operating systems and networks through one standard set of APIs. Even in this case, the traditional application development process requires writing code from scratch using third generation languages (3GLs) such as C or C++. Typically, this method of application development is used by horizontal application vendors who work on multi-million dollar projects that can take years to complete.

Corporate and vertical software developers, however, can no longer afford to create applications using this method. Through several corporate advisory councils, Novell has learned that desktop custom applications have about one-fifth the lifespan and take 50 percent more time to create than mainframe custom applications. Corporations cannot downsize their mainframe applications without a more efficient method of developing and deploying applications. As a result, corporations are running into severe roadblocks when they attempt to replicate mission-critical mainframe applications on desktops and networks.

The industry has reached a point where developers are demanding a

more effective response to these problems than today's tools and techniques can provide. Just as the operating system solved the application crisis for desktop computers, AppWare solves the current crisis for network applications (see Figure 1 on page 8 of the hard copy).

The AppWare Solution

AppWare is not an operating system or application. AppWare is a new layer of software that provides two components of technology to solve the two primary challenges described previously. These two components are the AppWare Foundation and the AppWare Bus.

The AppWare Foundation is a "fire wall" that insulates programmers from the growing complexities of multiple operating systems, GUIs and networks. The AppWare Foundation provides 3GL application programmers with a consistent, cross-platform set of APIs. By writing to this single API set, developers can access existing GUIs, operating systems and network services.

The second AppWare component, the AppWare Bus, provides large-grained, interchangeable software modules that corporate and vertical developers can use and reuse to quickly construct new network applications, without having to write code. These software components are called AppWare Loadable Modules (ALMs).

The AppWare Bus and ALMs are the software equivalent of the PC mother board and interchangeable plug-in cords. When ALMs are plugged into the AppWare Bus, their functionality becomes rapidly accessible for building new applications. In addition, third parties can offer their technology in the form of new ALMs.

The AppWare solution provides the following benefits:

- AppWare enables developers to easily access network services from a variety of desktop platforms. AppWare makes it as easy to incorporate messaging, telephony, multimedia, imaging and other network capabilities into an application as it is to incorporate basic file and print services.

- Traditional programmers can use one set of APIs, instead of having to master the underlying details of how to define, access and use the wide array of services available today. As a result, developers can concentrate on application-specific functionality.

- Reusing existing software modules greatly reduces application

development time. AppWare enables developers to construct applications with only a fraction of the time and effort required when building an infrastructure along with an application.

- Once familiar with AppWare, developers can reuse its components as needed, rather than having to recreate the same components for each new application.

- AppWare is open at all levels and will be available on all major operating systems and networks. Therefore, it opens broader application markets than have previously existed.

- Because AppWare is inherently multiplatform in its design and capabilities, it relieves developers of most of the effort required to build and maintain different application versions for each platform they support. Applications written on top of the AppWare Foundation API set can be recompiled to run on DOS, MS Windows, Windows/NT, OS/2, UnixWare and the Macintosh desktops, as well as NetWare and UnixWare servers.

Novell intends to work with developers, development tool vendors, hardware and operating system suppliers, and other third parties to make AppWare a standard for network application development. AppWare will remain open and extendible and will incorporate important development and interface standards. Therefore, AppWare can be welcomed into companies that must protect themselves from the vagaries of proprietary system components.

AppWare: A Technical Description

This section starts by describing the heterogenous nature of today's business environment which has led to the developmnet of AppWare. This description is followed by a representation of the AppWare components and the third-party tools that support it. The section concludes with an example of using AppWare for developing network appplications.

The Business Computing Environment

Contemporary computing environments consist of multiple hardware and operating system platforms. These platforms range from network or application servers, such as NetWare or UNIX, to mainframe systems that process transactions and act as repositories for consolidated enterprise information.

New facets of these environments include mobile computing and dedicated systems. Mobile computing encompasses notebook systems and personal digital devices, all of which need access to network

services and messaging support. Dedicated systems consist of microprocessors or computing systems embedded into a broad range of devices, from machines on the shop floor to hand-held data acquisition devices for inventory management in retail settings.

As microprocessor technology continues to decrease in cost, computing intelligence will find a role in almost every piece of machinery and in every appliance imaginable. Examples range from lathes on the shop floor to hand- held devices. The term ubiquitous computing applies to any intelligent device that incorporates some computer technology. To reap the maximum benefits from these intelligent devices, developers must include them in the networked environment so they can access, as well as provide, network services.

As Figure 2 shows (see page 11 of the hard copy), a heterogeneous computing environment clearly exists and includes the following:

- At the desktop, MS Windows, Windows/NT, Macintosh, OS/2, Novell DOS and UnixWare Personal Edition provide operating system services.

- At the server, NetWare and UnixWare provide network services that cover everything from basic file and print to database and telephony services.

- Heterogeneous networking environments include IBM's SAA, USL/SunSoft's Open Network Computing (ONC), the Open Software Foundation's (OSF) Distributed Computing Environment (DCE), as well as Novell's NetWare operating system.

- Mainframe and mini-computer platforms host legacy systems, high-volume transaction systems and an enterprise's repositories of consolidated information.

- Intelligent (or ubiquitous computing) devices such as manufacturing equipment on the shop floor can be integrated through the network with a CAD engineer's design workstation, so that they may be reconfigured electronically.

Figure 3 (see page 11 of the hard copy) shows the three categories of software required to connect these many different systems.

Each of these categories offers many competing APIs and services. Figure 4 (see page 12 of the hard copy) highlights some of these APIs and services.

In the network services category, a number of network services exist, including electronic messaging, database, directory services, and others. The complexity is intensified because many of these services have several available implementations, with different APIs. Consider electronic messaging, for example. At least four competing standards exist for messaging APIs: vendor independent messaging (VIM), Microsoft Messaging API (MAPI), CCITT Standard X.400 and Novell Message Handling Services (MHS).

In the client operating systems category, developers need to select the appropriate operating system for particular applications. Some of these operating systems include MS Windows, Macintosh, UNIX, OS/2 and DOS.

At the network category, multiple network standards can deliver network services to the layers above. Appendices A through D describe in detail the technologies that comprise this layer.

Each category offers a wide array of valuable choices. Each single API or standard can be a critical technology essential to the construction of a particular application. The challenge today is to leverage, unify and integrate these choices to produce real-world, mission-critical network applications. This is the infrastructure on which AppWare rests.

AppWare's Components

Just as a set of blueprints lays out the plans and components needed to construct a building, a computing architecture lays out the conceptual model required to organize a particular system. In addition, a computing architecture identifies the system's components and their relationships. Figure 5 (see page 13 of the hard copy) illustrates the components of AppWare.

AppWare is based on the concept of leveraging existing technologies. Thus, the AppWare architecture builds on top of all major client operating systems and most major distributed services, including file systems, shared printing resources, document management, imaging, telephony, digital multimedia and directory management services.

In the past, many of the functions these services supplied were hard-coded into applications. This doubled the effort required to deliver an application. Early GUI programs had to be built directly to the graphics hardware, which proved to be an extremely time-consuming task. Developers soon learned to create libraries of the common, reusable functions, leading to the development of specialized GUIs. Though applications with GUI interfaces were easy to use, their development

cost could still be prohibitive. It was not until the Apple Macintosh pioneered the availability of a widespread, consistent GUI, that applications built around this technology began to gain momentum.

Database technology development has experienced a similar progression. At first, developers wrote applications directly to the file system, and the data was managed differently for each individual application. The format of the data was also application-specific, which made it difficult to share data among applications. Again, the next stage was to construct common libraries for data storage and access. This refinement allowed particular systems of programs to share data. However, as data storage and access technology continued to become more widespread and complex, corporate MIS were unable to keep up with the need to deliver data over the network to multiple users across multiple platforms. In response, database vendors began providing cross-platform database engines that a number of applications share over the network. Today, those same database vendors are trying to provide standards, such as Structured Query Language (SQL) and Integrated Database API (IDAPI), that address the issues inherent in managing data simultaneously across multiple databases and data models.

A broad variety of vendors providing a wide array of services, as well as multiple implementations of these services, has created the need for the first AppWare component, called the AppWare Foundation.

The AppWare Foundation

The AppWare Foundation hides the complexities of networks, local operating systems, and GUIs from developers who write line-by-line code using 3GLs, such as C, C++, COBOL and Pascal (see Figure 6 on page 14 of the hard copy). The AppWare Foundation provides a consistent, standard set of APIs that allows developers to access the services provided by both the network and the local operating system. Using the AppWare Foundation, developers can create cross-platform network applications without compromising system performance, application functionality and compatibility with existing and emerging technologies.

The AppWare Foundation includes the Universal Component System (UCS) technology which Novell acquired with the purchase of Software Transformation, Inc. Other components include the CPI-C interfaces for host connectivity and the X/Open distributed transaction processing APIs, which are supported by Tuxedo. The AppWare Foundation also supports multiple compound document architectures, such as Apple's Compound Document Architecture and Microsoft's Object Linking and

Embedding (OLE) architecture. Appendix E describes these technologies.

Applications written to the AppWare Foundation can connect transparently to existing network services. An application programmer need no longer be aware of where a service is located or how it can be accessed. The programmer simply chooses the application's target platform, and the AppWare Foundation builds the necessary code to connect the application to the native services of that platform. When the developer recompiles the same code for a different platform, the AppWare Foundation implements the appropriate native services for the chosen platform, again without involving the programmer. For example, using the AppWare Foundation, a programmer writes the same code whether he or she is accessing files from an application on a Macintosh, MS Windows or a UNIX platform.

Application Portability

Using the AppWare Foundation's API set, the developer writes the application code only once and recompiles it to run on multiple platforms. This kind of portability for desktop applications is critical because many organizations have a mix of desktop platforms. Even organizations that have settled on a single desktop platform may need such portability to accommodate and take advantage of new technologies as they emerge. The AppWare Foundation provides portability across a wide variety of platforms, letting businesses take advantage of new technologies while preserving current system investments.

The goal of platform portability toolkits is to provide the same functionality across different platforms. As simple as this may sound, it is actually quite difficult. For example, almost all applications allow users to enter text in one form or another. However, text-editing ranges from basic editing, such as modifying the data fields in a forms package, to highly sophisticated editing, such as revising a document using a publishing system. When features are available on all platforms, as is the case with simple text entry fields or pull-down menus, compatibility problems seldom arise. The real question is how a multiplatform development environment should handle features not available on all platforms, such as multiple fonts in a text box or undo capability.

Platform portability toolkits typically take one of two approaches: they provide only common features (least-common-denominator toolkits) or they provide all the features (superset functionality). Taking the least-common-denominator approach is very efficient, but does not meet

specific functionality requirements. However, providing a complete superset of the features available on each platform is probably impossible.

The AppWare Foundation does not provide portability by dropping to the lowest common denominator of all the supported environments. Rather, it identifies real-world application requirements for given areas of functionality. Upon establishing such requirements, functionality was added to those platforms that required it. While this is a superset approach, AppWare Foundation stops short of providing a 100 percent superset, concentrating instead on the functionality that commercial developers are most likely to need. Feedback from current users indicates that the AppWare Foundation is one of the most robust technologies available.

Most horizontal business applications can be coded exclusively to the AppWare Foundation, ensuring maximum portability and transparency of access to network services. However, the architecture is flexible and will allow programmers to drop to the operating systems' native APIs to gain more direct access to the system software or hardware as needed. Therefore, applications can still take full advantage of the unique functionalities and native services of local operating systems and networks.

In addition, programmers are free to use the platforms, compilers, linkers and debuggers of their choice. The AppWare Foundation is compatible with Symantec, Borland, Microsoft, MPW, Lightspeed, SABER and GNU compilers, and with Multiscope, Codeview, and SADE debuggers.

Underneath the AppWare Foundation exists the Common Request Broker Architecture (CORBA) specified by the Object Management Group (OMG). CORBA provides an infrastructure that allows objects to communicate, and is independent of specific platforms and languages used in the implementation of the objects. The CORBA architecture is described in Appendix F.

Benefits of the AppWare Foundation

The AppWare Foundation provides application developers with a complete set of APIs for implementing enterprise business applications using 3GLs. The benefits of the AppWare Foundation include the following:

- A single API set for different operating systems and networks

- Portability of the applications built to the AppWare Foundation
- Transparent access to network services
- High application performance
- Support for the evolution to distributed objects

For additional information about the AppWare Foundation, please refer to the Novell AppWare Foundation White Paper.

The AppWare Bus and AppWare Loadable Modules

Although the AppWare Foundation simplifies the application development process, it supports only those developers who write applications with 3GLs. In other words, one must still be a well-trained programmer in traditional programming languages to create applications based on the AppWare Foundation set of APIs. Corporate and vertical software developers using 4GL and 5GL tools require a much more rapid, efficient development platform.

In these rapidly changing environments, access to prefabricated software is the key, thereby transforming the creation of applications from an art form to an assembly line production model. Building software with 3GLs is similar to building a car from raw metal, glass, rubber, and plastic. Building software using reusable modules is parallel to building a car from ready-made parts, such as an engine, wheels, seats, instruments and a steering wheel.

The AppWare Bus is a software engine that does for applications what the hardware bus did for personal computers -- namely, it manages and coordinates the interaction of prefabricated, plug-in software components called AppWare Loadable Modules (ALMs) (see Figure 7 on page 17 of the hard copy).

ALMs are software objects that provide access to the functionality provided by both local operating systems and network services. ALMs can range from simple graphical utilities and spreadsheet modules to network services such as database and messaging. Business application developers can create new applications quickly and easily by linking different ALMs. The Novell Visual AppBuilder tool is designed specifically for this task. This tool is described in more detail in the following pages. Developers can also use other 4GL and 5GL tools that are compatible with the AppWare Bus to create new applications.

ALMs are large-grained, high-level software objects, which means they

are much larger and more automatic than, for example, C++ classes. In a typical business application, a developer may use only 25 different ALMs to create all the needed functionality. Using C++ classes, as many as 500 to 1000 different classes can be used to create the same functionality. ALMs are more intelligent and functional than lower-level software components such as classes, but are smaller than today's massive horizontal applications (See Figure 8 on page 17 of the hard copy). Since ALMs plug into the AppWare Bus, they can communicate and work together -- even when created by different programmers.

Corporate developers typically have many in-house projects with overlapping components, such as custom reporting applications for different departments that must access the same data. Using ALMs, corporate developers can link the appropriate modules to create applications, reusing existing modules and leveraging each other's work.

Creating ALMs

Novell has already created a number of basic and network ALMs, such as MHS and Btrieve, and is actively working with a number of ISVs to provide additional ALMs. For example, Cheyenne Software is creating ALMs for imaging and document management. Various third parties have already built ALMs for accessing Oracle and Sybase databases. In addition, MIS programmers may want to create new ALMs to provide key elements of functionality with information systems. Then, business application developers can link ALMs without having to understand how they were built.

ALMs are built using 3GLs and the AppWare Foundation. ALMs can be created with almost any Microsoft, Borland, Symantec, or MPW compiler. The Novell ALM Construction Kit provides the interfaces necessary to plug ALMs into the AppWare Bus.

AppWare eliminates at least two risks for custom software providers: time-to-market and environment selection. The ALMs available on the AppWare Bus can significantly reduce application development cycles. Also, AppWare's availability for most major commercial desktops reduces the cost of a porting effort to a simple recompile.

Novell Visual AppBuilder

Novell now offers a high-level programming tool that has been tightly integrated into AppWare. The Visual AppBuilder tool provides an environment for rapid application development of network and standalone applications. Visual AppBuilder allows applications to be

constructed by selecting icons that represent different ALMs. Then, developers use on-screen links to create application logic (see Figure 9 on page 18 of the hard copy). Visual AppBuilder provides a 5GL-level development environment, enabling programmers who do not necessarily know the details of 3GL development to rapidly create full-featured, reliable applications.

The range of applications developers can create is limited only by the range of ALMs provided by Novell and third parties. This makes Visual AppBuilder an ideal, easy-to-use tool for developing corporate and vertical business applications that keep pace with today's changing business environment.

For example, consider a developer who wants to provide an easy-to-use automated telephone directory system for a global corporation. Suppose the directory information is stored in an Oracle database and the users of this program are non-technical, so the application needs a multimedia front end. In addition, the available hardware offers automatic answer and dialing through the telephony services on the network.

Using a 3GL tool, it would take several years and many developers to create such an application. Visual AppBuilder enables a few developers to build this application in only a week. A developer would simply connect graphically the Oracle database ALM, the multimedia ALM and the telephony ALM to create the new application without writing traditional line-by-line coding. Because ALMs are based on the AppWare Foundation, the application can be ported quickly to any common operating system and network.

ALMs make available all the services provided by the network and the local operating system. With ALMs for directory services, security, licensing, software distribution, telephony, work flow, imaging, multimedia and even office productivity, a new world of networked applications is now open to corporate and vertical software developers.

Benefits of the AppWare Bus and ALMs

The AppWare Bus, ALMs and Novell's Visual AppBuilder benefit corporate developers, smaller ISVs, system integrators and Value-Added Resellers (VARs) whose software needs exceed horizontal productivity applications. Some of these benefits include the following:

- Rapid "plug-and-play" model for application development
- Network services available to all developers

- Reliable, powerful applications for corporate MIS, vertical software developers and system integrators

- 4GL and 5GL development tools, rather than 3GL tools

- Use of 100 percent open and extendible through new ALMs

- Portable over major operating systems and networks

For additional information on the AppWare Bus, ALMs and Novell's Visual AppBuilder, please refer to the Novell Visual AppBuilder White Paper.

Integration with Third-Party Tools

AppWare's open architecture allows application developers to integrate the AppWare Foundation and AppWare Bus with a variety of third-party tools to provide additional functionality. For example, using AppWare Foundation APIs with Microsoft's Visual C++ can increase the efficiency of creating cross-platform applications. Novell's goal is to work closely with third party tool vendors to accomplish the following:

- Provide access to network and other services through the AppWare Foundation, so developers can deliver more sophisticated applications while writing less code.

- Extend the base of ALMs, allowing developers to quickly construct business solutions by combining ALM building blocks.

- Allow 4GL tool vendors, such as Gupta Technologies, Powersoft and Easel Corporation, to access the functionalities of ALMs.

Using AppWare: An Example

Suppose a developer wishes to create a networked document management application that will, among other things, route documents to fax servers and electronic mail servers. The application is designed such that the document destination handling is performed on a file server, but the user interface and editing are handled at the client workstation. Consider the development process using three different approaches of software development tools.

First Scenario

Using a relatively traditional set of application development tools, a

developer must write core application modules that perform the document editing, as well as other software modules. Developers would have to do the following:

- Design protocols to establish communications between the client workstation and the machine providing the routing services

- Handle the construction, buffering, and sending and receiving packets at both ends of this process.

- Handle error detection, resending and acknowledging packets.

- Design some mechanism for locating the service provider, perhaps as rudimentary as having the user enter the name or address of the remote machine.

- Design an algorithm for establishing and verifying the identity of the client, possibly involving maintaining passwords or keys.

- Design and implement a minimal command language to enable the processes to perform such functions as opening a file remotely (for reading and writing).

- Design a user interface and implement the routing services on the target server machine and design the user interface code on the target client machine.

- Port the code for both the client and the server to several different environments, each one having specific communications code, graphics code and operating system calls.

This scenario provides innumerable challenges to the programmers and takes months or years to implement. It also presents a challenge for testing and maintenance, which grows exponentially with each new version of the product.

Second Scenario

In the second scenario, the developer of the same document management application will still have the same editing and routing code to implement, but can limit the communications and housekeeping modules. In this scenario, the developer may use, for example, Remote Procedure Calls (RPCs) to establish communication and perform the authentication, RPCbind or NIS to locate the service, and some GUI tools to build the user interface. The developer must write a specification file for RPCgen, determine whether a datagram or

guaranteed message service is necessary for this specific application, and provide a means to specify the acceptable transport service, such as a configuration file. After writing the service code for the server and the user interface and writing the front-end code for the client workstation, the developer ports the code to different platforms, such as Macintosh, MSWindows and UnixWare.

Third Scenario

In the third scenario, the developer uses Visual AppBuilder or another AppWare-compatible tool to rapidly build his application, on both the server and the client sides. The developer designs and implements the solution by graphically linking ALMs. The ALMs, for example, might be a text editor, a document tagging service, fax services and electronic mail services. The ALMs are represented by icons and the application is created by linking the appropriate icons.

The ALMs the developer uses have already been written to the AppWare Foundation. These ALMs rely on the underlying location broker to locate the fax and e-mail services, to verify and select the appropriate available transports, and to authenticate the user. The developer uses the windowing system provided by GUI ALMs to build the user interface and recompile the application once for each target environment.

Conclusion

Building an infrastructure is an ambitious undertaking which can never be totally completed. The construction of an interstate highway system provides a good example. The expense and effort were considerable, but the benefits reaped from making new businesses and opportunities possible far surpassed the outlay in resources.

Similarly, a development infrastructure like AppWare can enable applications that might otherwise be impossible to bring to market. By providing an underlying collection of high-level, advanced services and the communications necessary to link service consumers and service producers, AppWare makes the construction of distributed applications far simpler than ever before.

Instead of the five- to eight-year effort it has taken to bring groupware products to market, AppWare can cut the time required to deliver such products to less than two years. The time savings results from the ability to use predefined services for database, calendaring, messaging and the like, without having to construct them from scratch. Also, the ability to use the network without having to subdue the protocol,

communications and handshaking issues saves additional time.

By adopting AppWare as a development framework, application developers gain more than time. They gain:

- Access to a broad range of platforms without requiring multiple instances of their code for every platform they wish to support, eliminating the support and maintenance headaches.

- Access to a rich set of services along with transparent network access, thereby cutting application development time and extending the kinds of application problems they can solve.

- Control over a set of application tools and methods enabling developers to easily create client- and server-side modules.

All this adds up to increased developer productivity, better application flexibility, improved leverage of effort and broader markets for software.

By incorporating AppWare-based applications into their networks, users benefit from AppWare. Users gain:

- More broadly available network services. The power and flexibility of AppWare-based applications will make distributed services more broadly available.

- More flexibility and ability to change in the face of shifting needs and requirements. The customized nature of AppWare-based applications enables this flexibility.

- More extendible applications. The open-ended nature of ALMs makes such applications far more extendible than before and enables them to take advantage of new or enhanced services.

All this adds up to increased user productivity, more responsive and capable applications and rapid incorporation of technological changes and advances.

Why Novell?

The primary reason Novell is taking this bold step is to break the application backlog that threatens to slow the growth of the networking industry. Other important reasons include responding to customers' demands for more advanced, distributed applications; the desire to make networking completely central to information systems;

and the fundamental need to improve the technology used to solve business problems. Novell believes its work and expertise in laying the groundwork for an application development infrastructure, and its openness to working with partners and customers, make Novell uniquely qualified to meet this challenge.

Although Novell did not focus on the development marketplace in the past, it is uniquely positioned to deliver a development infrastructure. For the last ten years, Novell has been building some of the most advanced networking solutions available in the marketplace. Novell offers support for more networking topologies and technologies than any other vendor. Likewise, Novell supports more client and service platforms than all other vendors, ranging from desktops to mainframes, all of which can interoperate freely within the NetWare environment.

Novell also offers the broadest range of integrated internetworking solutions, for both wide-area and local-area access, across most available communications technologies. These offerings comprise the most complete communications and interoperability solution available from any vendor. In summary, Novell has the networking and interoperability coverage needed to support an infrastructure.

Just as Novell has shielded complexity and raised the level of network productivity in the past, Novell can shield complexity and raise the level of productivity for networked applications development and deployment.

No single company can provide a complete infrastructure, and Novell's unparalleled partnerships and programs play an important role in the development of a complete infrastructure. Novell has always been a partnering company; Chairman and CEO Ray Noorda is credited with coining the term "coopetition" to reflect Novell's willingness to cooperate with its fiercest competitors. Novell has also been an impetus for forming industry-wide groups, like the Technical Support Alliance (TSA), to bring vendors together under a single support umbrella, and avoid the finger-pointing exercises that network troubleshooting can so often cause.

Novell offers testing and compliance programs, such as "Yes It Runs With NetWare" for NetWare-compatible products. Novell also has one of the broadest education and certification programs for networking specialists in the world, the "Certified NetWare" professional programs. In addition, Novell offers options for training, service and support that continue to be widely emulated throughout the computing industry. Novell also maintains relationships and alliances with key consulting

firms, major platform vendors and customer groups, to stay in close touch with industry trends, technologies and customer requirements. These partnerships and programs demonstrate Novell's ability to enlist broad support for an infrastructure, as it works closely with all the key players needed to stay ahead of emerging technologies.

In addition, Novell offers a wide range of distributed services to multiple client platforms through its NetWare and Unix products, from file and print, to database, messaging and directory. In the next 18 to 24 months, Novell will add support for services to provide electronic software distribution; software license metering and monitoring; imaging; telephony; document management; and multimedia from internal efforts and with joint development with companies like Imagery and Fluent Technologies (multimedia). These services highlight the ideal platform Novell offers to supply the advanced services that an infrastructure can deliver.

Novell has the ability to excel at enterprise networking and interoperability; it has the partnerships and programs needed to support enterprises; and it provides the richest set of distributed services. Therefore, Novell is pushing forward to build an application development infrastructure to leverage these assets.

Novell clearly recognizes that the process of realizing the AppWare architecture is a lengthy, difficult and labor-intensive task. Novell cannot tackle this effort alone; significant input and cooperation from partners and developers has been and will be required. Even so, the potential benefits are enormous, and the rewards are significant.

To prove its depth of commitment to AppWare, Novell will publish the AppWare interfaces as they are defined. Novell also intends to keep the AppWare development tools open to all interested third parties, and will augment the technologies and interfaces that AppWare accommodates based on customer requests and market demands. In addition, Novell is committed to using AppWare for its own internal development efforts.

Novell is committed to providing complete education, service and support for all components of the AppWare initiative, and to providing copious background and training materials about the AppWare architecture. Since the value of an infrastructure is measured only by the way it is used, Novell's primary goal is to build an infrastructure that suits the needs of the developers who use it.