

Windows and UNIX on the Desktop

MetaFrame gives you the best of both worlds

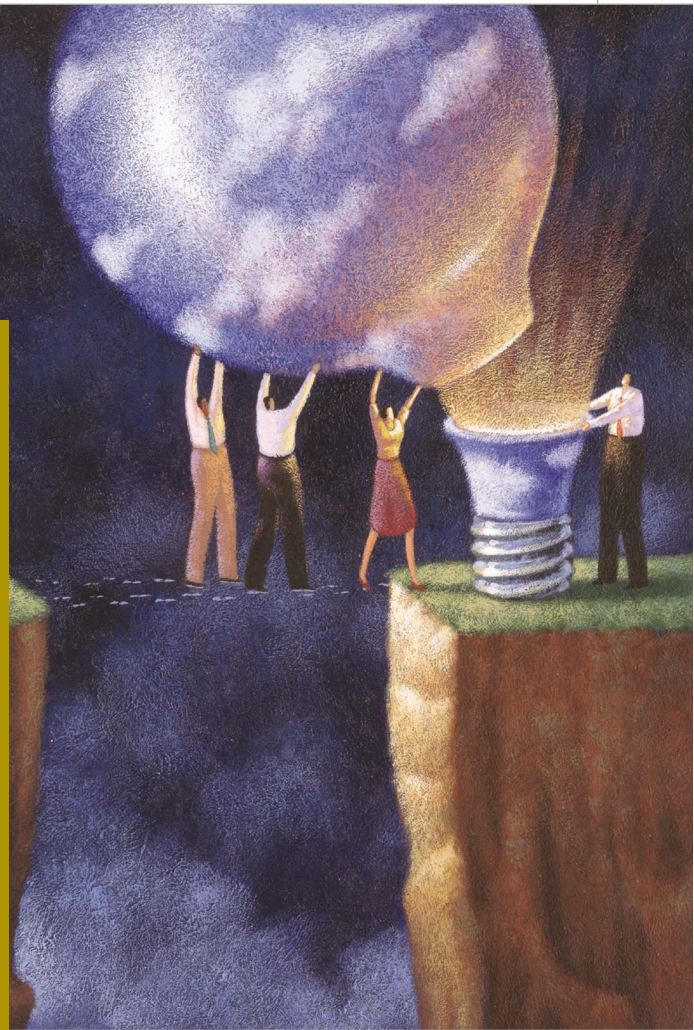
A key benefit of server-based computing is that with minimal effort, you can use the technology to provide seamless interoperability. For example, instead of running terminal emulation software to make one platform act like another, you can run display software on a client machine and display all graphical output from an application that's running on the terminal server. This scenario probably isn't news to you—a lot of people use terminal services to run Windows applications on non-Windows platforms. But what might be news to you is that you can employ the same technique to get UNIX applications to non-UNIX platforms without an X Window System (X) client. In addition, Citrix MetaFrame's ICA can make UNIX and Windows applications available from the same interface and enable you to copy data between the two kinds of applications. Let's examine how to set up MetaFrame for Windows and MetaFrame for UNIX to present Windows and UNIX applications from one UI, taking into account the challenges of licensing and configuring logons. I assume that you've already installed MetaFrame and Citrix NFuse, which lets you deploy traditional client/server applications over the Web, and are prepared to publish applications.

Why MetaFrame for UNIX?

Citrix originally developed its multiuser product for the Windows platform, but MetaFrame has been available for UNIX since 2000. Like MetaFrame for Windows, MetaFrame for UNIX uses the ICA protocol to display applications on user desktops.

CHRISTA ANDERSON
(christa@winnetmag.com) is a senior contributing editor for *Windows & .NET Magazine* and senior technologist for triCerat Software. Her most recent book is *Windows Terminal Services* (Sybex).

STEVE GREENBERG
(steveg@thinclient.net) is the founder and president of Thin Client Computing in Scottsdale, Arizona. He has designed mission-critical solutions for various Fortune 500 companies.



UNIX has supported multiple users for decades, enabling any computer with an X client to display applications running on a separate UNIX machine. So why do you need more software to achieve something that you can do already? The simple answer is that ICA lets you do something that you can't do with X.

Until recently, the ICA display environment wasn't as rich as X's and its more powerful graphical rendering machine. However, ICA has its advantages. First, the ICA client component requires almost no configuration, regardless of the application you're running on it. In contrast, X emulation packages for non-UNIX clients often require a significant amount of configuration and tuning to work optimally. Second, the ICA protocol displays objects extremely well over low-bandwidth connections, whereas X requires too much bandwidth to work effectively over dial-up connections and most WAN connections. Third, ICA supports session shadowing, which lets one person see and interact with another's terminal session. Finally, ICA completely isolates the display component of an application from the application's execution, so you can disconnect one computer from a running terminal session, then reconnect to the session from another computer to pick up where you left off. Meanwhile, the application on the server retains the session data in memory. This feature alone has made MetaFrame for UNIX popular among UNIX developers because it lets them end a session at work and reconnect to it from home.

ICA's simpler software on the display device and lower bandwidth requirement result from differences in the way X and ICA work. In X, the display device is considered the server, and the host running the application is considered the client. This arrangement might seem backward at first glance, but if you think about it, it makes sense. The device that displays the application is a display server (i.e., the graphics rendering system). Graphical primitives travel from the UNIX application host (the client) to the display device (the server) for local rendering and display. In ICA, the display device is the client and the application host is the server. The server renders the graphics, then transmits them over the wire to the client for display.

MetaFrame for UNIX isn't as mature a product as MetaFrame for Windows. MetaFrame for UNIX 1.1 (the current version) roughly corresponds to MetaFrame for Windows 1.8, and MetaFrame XP—essentially, 2.0—is the current version. MetaFrame XP for Windows has some scalability and printer driver-management

tools that earlier versions lack and implements the new Independent Management Architecture (IMA).

MetaFrame for UNIX doesn't support the concept of a server farm. You can't make MetaFrame for UNIX servers part of a MetaFrame for Windows server farm, nor can you place the MetaFrame for UNIX servers into a farm of their own. MetaFrame for UNIX does let you publish applications from MetaFrame for UNIX servers; define access for users, groups, and folders; and perform load balancing. Citrix says that MetaFrame XP for Windows' graphical Java console will be available for UNIX "in the future," but you must perform all MetaFrame for UNIX configuration from a command line until then. However, MetaFrame for UNIX is on the same development track as MetaFrame for Windows, and Citrix designed the two products to work together. In fact, one reason why MetaFrame XP for Windows has a Java-based management console (instead of a Microsoft Management Console—MMC—snap-in) is so that you can use the same administrative tools for the two products. (That said, the speed of using the command line might make you wish you could publish applications from the command line with MetaFrame for Windows.)

Creating the Integrated Environment

The first step in creating an integrated environment entails making UNIX and Windows applications available to the same UI. If you already use MetaFrame for Windows and want to add MetaFrame for UNIX to the mix, you'll be glad to know that the ICA

client that your users use to connect to MetaFrame for Windows servers will also let them connect to MetaFrame for UNIX servers. To publish an application from MetaFrame for UNIX, you must use the `ctxappcfg` command-line utility, not the Published Application Manager that you use for MetaFrame for Windows. Log on as superuser or as `ctxsrvr`, the designated Citrix administrator account, then walk

through the following process (this sample exchange publishes `xterm`, a terminal command-line window, from a Hewlett-Packard HP-UX server running MetaFrame for UNIX):

```
# /opt/CTXSmf/sbin/ctxappcfg
App Config> add
Name: XTERM
Command Line: /usr/bin/X11/
xterm
Working directory:
Anonymous [yes|no]: no
Successfully added configuration
for "XTERM".
App Config> exit
```

MetaFrame for Windows administrators working with MetaFrame for UNIX for the first time will notice some differences. First, although you can use the Administrator account to log on to MetaFrame for Windows, by default you can't use the root account to log on to a MetaFrame for UNIX session. (This limitation is a generally accepted security restriction in UNIX, which doesn't let you log on as root from anywhere but the console.) Second, you must apply Feature Release 1 (FR1) and any available patches to the UNIX servers before publishing applications. FR1 gives your terminal sessions higher color depth (up to True Color) and adds some other features, such as multiple-monitor support and application-specific fixes. Until I applied

Adding ICA support to UNIX lets it do things that aren't possible with the X protocol.

the patches to our Sun Microsystems Solaris server, applications that were supposed to appear in seamless windows came up with a small blank desktop window, which I had to close.

When the applications are available, Win32 clients can connect to both kinds of MetaFrame terminal servers from the same tool. From the Citrix Program Neighborhood (Start, Programs, Citrix ICA

Client), click Application Set Manager, then choose Custom ICA Connections. When you start the wizard this way, you can browse for both Windows and UNIX servers, as Figure 1 shows. The list that appears displays the servers; select Published Application instead of Application Set Manager to see the applications published from both Windows and UNIX servers. When a user creates connections to any published application or server desktop, those connections will appear in the Custom ICA Connections group.

NFuse

MetaFrame has long supported a Program Neighborhood for Win32 clients. This feature lets you display icons for published applications in a dialog box on the desktop (or even pushed to a user's Start menu or Desktop). However, for a long time, this Program Neighborhood wasn't available to non-Win32 clients. Even on platforms that do support Program Neighborhood, you could create custom ICA connections or connect to one Citrix farm, but you couldn't browse across farms. In 2000, Citrix introduced NFuse, a Web-based Program Neighborhood that lets you see application icons from any farm in a Web browser and launch applications from those icons, assuming you have the appropriate permissions. To further streamline application connectivity, you sign on to this application Web page just once to access all applications from any farm.

To make application icons appear in Web browsers, Citrix relies on three technologies—MetaFrame, a Web server, and XML. The MetaFrame servers can be Windows- or UNIX-based, and the Web servers can be Microsoft IIS or Apache. The XML services, collectively called NFuse and available for free on the Citrix Web site, enable the MetaFrame servers and the Web servers to communicate.

To determine who can see which applications in MetaFrame for Windows, you use Published Application Manager. In MetaFrame for UNIX, you must publish the applications, then separately filter access to the published applications by user or group. (MetaFrame for UNIX filters apply only to applications that NFuse presents. The ICA client browser displays all published applications, regardless of

the filters.) When a user loads the NFuse logon Web page, he or she must enter a username and password. If the user is authorized to log on, NFuse displays the suite of applications he or she is permitted to access.

When the user launches an application from the NFuse page, the local ICA client launches and accesses the MetaFrame server that publishes the application. Thereafter, all communication occurs between the ICA client and the MetaFrame server directly—the Web server is no longer involved; its role is simply to present the link to the application on the MetaFrame server.

If you want to publish both Windows and UNIX applications on one Web page, you might also want to let users log on once to gain access to both servers. The easiest way to configure NFuse for one logon is to give each user one username and password for both the Windows and UNIX terminal servers.

An alternative is to share the security databases so that you don't have to create special NFuse accounts to access both application types. To share the databases, you must employ some Windows-UNIX authentication integration tool—for example, Microsoft Services for UNIX (SFU).

How Licensing Works

When you're working with both MetaFrame for Windows and MetaFrame for UNIX, licensing can become tricky. MetaFrame for Windows requires both per-seat Terminal Server Client Access Licenses (TSCALs) because it runs on Windows 2000 Server Terminal Services and per-connection MetaFrame licenses. The per-seat licenses are in addition to the CALs required to connect to any Windows server. MetaFrame for UNIX, in contrast, requires only the same per-connection MetaFrame licenses that MetaFrame for Windows uses.

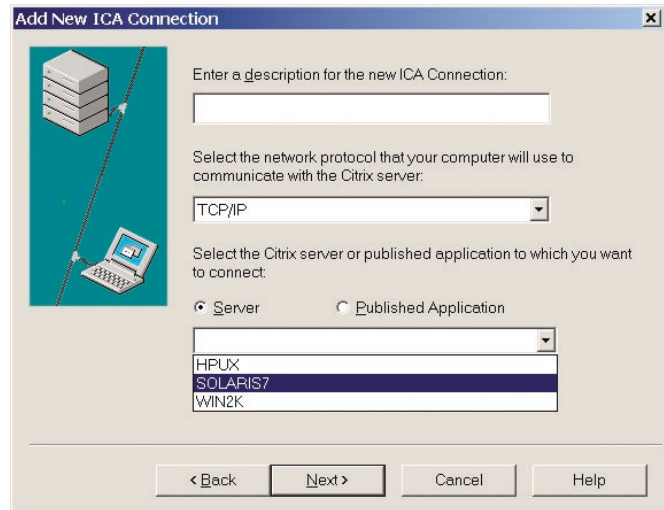



Figure 1: Browsing for Windows and UNIX servers

In fact, Windows and UNIX terminal servers running MetaFrame 1.x can share the same license pool. UNIX doesn't require any client licenses for the OS.

The Windows Terminal Services underlying MetaFrame for Windows uses a per-seat TSCAL explicitly assigned to the client machine that accesses the terminal server. Therefore, you must configure published Windows applications to permit access to only the people who need the applications. Otherwise, you'll lose TSCALs to anyone who idly clicks the application links.

Some old UNIX hands are apt to scoff at MetaFrame for UNIX as a way to make UNIX do something that it does already, but adding ICA support to UNIX lets it do things that aren't possible with the X protocol. ICA support lets you enjoy low-bandwidth access to graphics-intensive UNIX applications, deliver UNIX applications to platforms that don't support X natively, provide one interface for UNIX and Windows applications (with optional access from a Web interface), use Session Shadowing, and disconnect and reconnect to desktop sessions or published applications at will. In short, ICA and X are different protocols with different applications. ICA gives you true interoperability, making Windows and UNIX applications available to any user with an ICA client. Using ICA with Windows and UNIX lets you run your applications where they belong, instead of trying to retrofit them for platforms for which they're not designed. 

InstantDoc ID 38495