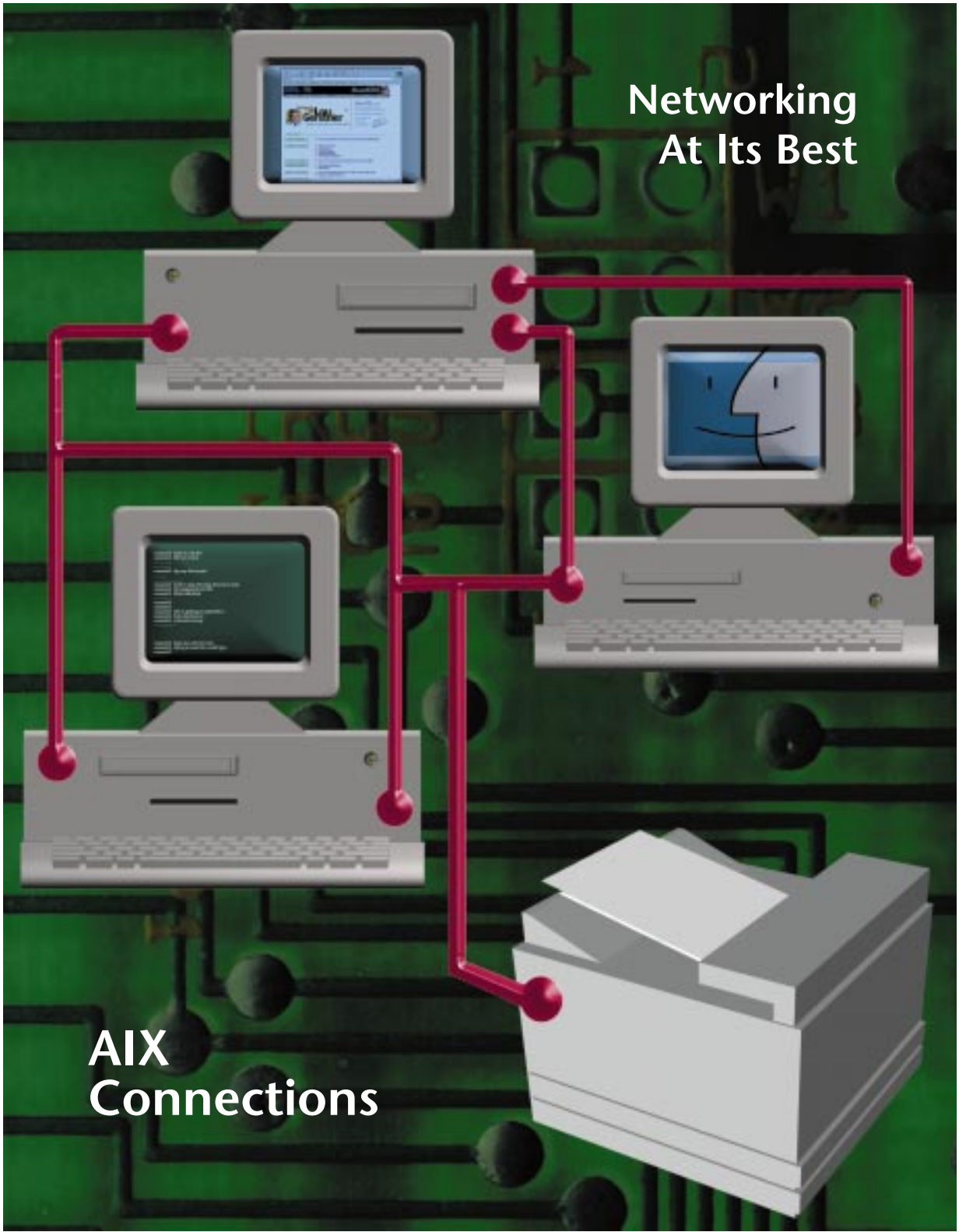
IBM's Magazine for AIX Developers June 1997

AIXpert

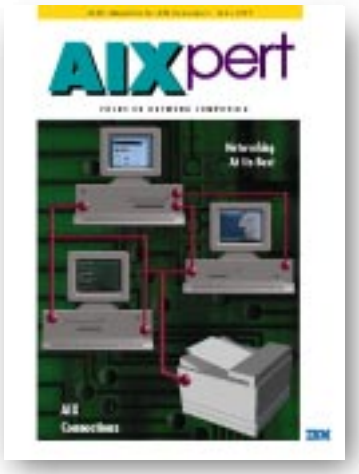
FOCUS ON NETWORK COMPUTING



IBM

AIXpert DIGITAL VERSION IBM

TABLE OF CONTENTS



Click here to view a full-size version of this issue's cover.

Click here to go to a complete version of this issue for printing.

JUNE 1997

AIXpert BACK TO MAIN MENU

Gentle(wo)men, start your servers!

COMMENTARY

If you've been skeptical about network computing as the next productivity leap in computing power and functionality, you can remain ambivalent no longer. Almost daily, new products enter the market that explore the power of this new computing paradigm—Web-based applications written in Java, new networking hardware to make access quicker and easier, rock-solid (*yes!*) security packages that protect the integrity of transactions across the Internet and even onto the user's desktop—the onrush of new ideas is endless! If you were at the recent IBM Technical Interchange in St. Louis (and if not, why not?), you had the opportunity to see and hear first hand every aspect of the network computing wave—classes to teach fundamentals and nuances, vendors demonstrating their latest products. You could even get connected yourself at the many Domino-powered network stations at the conference.

Within that framework, we've put together an issue packed with networking technology for you to sample. It's technology you can put to work for your customers' and your own productivity. AIX Connections, a feature of AIX that allows seamless interconnection of disparate computing platforms, is one such technology. First covered by *AIXpert* in our November 1995 issue, AIX Connections has some new capabilities of which you will want to take immediate advantage. Two articles in this issue highlight PC security and the new features in release 2. Another networking-based article describes the new IBM Network Station and how it fits easily into an AIX environment.

Interested in setting up a networking environment? Learn from the insights contained in the "Testing Cluster Solutions" article that puts many networking applications through their paces in a clustered-server environment. A server is worth little if it is not secure, so you'll be happy (though not surprised) to learn that AIX has earned a new European security certificate. Read all about it in "AIX Security Certification."

Our networking issue would not be complete without a Java article, and with "A Cup of Java Code" our cup overflows. Our new CD-ROM format allows us to include this Java tutorial article complete with the entire Java sample code that supports the article. Cut, paste, and work right along with the author to learn the ins and outs of Java.

Other interesting articles round out this issue, plus the popular "AIX Questions" feature. Be sure to check out all of the articles for information that you can apply to your situation.

While we're talking about networking, come visit us in our other incarnation. We've just revamped the Web site with a new high-tech look and more efficient navigational paths—the IBM Solution Developer Program Web site at http://www.developer.ibm.com/. See you there!

George Noren

George Noren, *IBM Corporation, Internal Zip 1034, 11400 Burnet Road, Austin, TX 78758. Internet: geo@austin.ibm.com. Since joining IBM in September 1979, Mr. Noren has written hardware and software manuals, including AIX and RS/6000 manuals, and was a member of the InfoExplorer™ design team. He has worked as a system administrator for several AIX systems and is a Certified AIX System Administrator. He is currently Editor in Chief of the World Wide Web site for IBM's Solution Developer Program (www.developer.ibm.com) in addition to his work with* AIXpert *Magazine. Mr. Noren studied engineering at Illinois Institute of Technology, holds a BA in English from the University of Minnesota and an MBA from St. Edwards University in Austin.*

George Noren

AIX Connections Release 2

By Denise Genty, Michael Lew, and Rakesh Sharma

The AIX Connections feature of AIX has a single goal: to provide the premier solution for integrating IBM-compatible and Macintosh® personal computers with UNIX solutions. AIX Connections allows files and printers to be shared by different networked PCs.[1]

AIX Connections Release 2 integrates the functionality of three separate servers from the initial release into a uniform framework of commands, realms, services, and interfaces. From a hierarchal perspective, AIX Connections is based on several different object types. The highest object level, the system, encompasses all of the AIX Connections components: the three realms, file and print objects, and different types of interfaces, as shown in Figure 1.

The three realms, based on the type of service offered, relate to three servers in Release 1, as shown in Figure 2. Most of the configuration and administration of AIX Connections is based on the realm type.

In addition to these realms, the Client component enables an AIX® workstation to behave as a network client so that AIX users can mount remote Server Message Block (SMB)-compatible and NetWare®-compatible volumes and share resources. The clients can access and view services, such as file,

Denise Genty

Michael Lew

Rakesh Sharma

The Hierarchy			Command
System			tnsystem
	Realm		tnrealm
		Service Type	tnstype
		Service	tnservice
		Volume References	tnvref
		Printer References	tnpref
		Attach Points	tnattach
		Client	tncdct
	Transports		tntransport
		Interfaces	tniface
	Volumes		tnvolume
	Printers		tnprinter
	Users		tnuser, tnpasswd

Figure 1. Object hierarchy

[1] For information on the technical aspects of AIX Connections Release 1, see "AIX Connections" by Bret R. Olszewski and Kay Chang in *AIXpert*, November 1995.

AIX Connections Servers and Realms

Server	Realm
Lsserver	NetBIOS (NB) clients
NWserver	NetWare® (NW) clients
Macserver	AppleTalk® (AT) clients

Figure 2. AIX Connections servers and realms

terminal, print, and NetWare Virtual Terminal (NVT). Either the AIX machine or the client can configure and administer these services. Clients also can locate each individual service by name.

A system may have multiple services defined and available to clients. Services have many attributes, such as command, path, encryption, depending on the type of service. Some service examples include the following:

- Transports are low-level networking protocol suites, such as NetBIOS or IPX/SPX. Transports contain interfaces and represent a set of networking interfaces used by each realm.
- A Volume is a directory on the UNIX file system that can be exported through a defined file service.
- A Printer is a UNIX print queue that can be exported through file and print services.

New Features in Release 2

AIX Connections Release 2 provides interoperability with existing AIX features, such as Distributed Computing Environment (DCE) and High Availability Clustered Multiprocessing (HACMP), plus enhanced system administration and configuration capabilities. New features include the following:

- DCE integration
- HACMP support
- Forward (Proxy) Authentication
- OS/2® extended attributes support
- Token Ring for the AT realm
- Web-based administration tool
- System Management Interface Tool (SMIT) enhancements
- NetBIOS WINS server
- TotalPrint print queue enhancements

DCE Integration

AIX Connections Release 2 includes integration with DCE and the Distributed File System (DFS) for the NetBIOS realm. DCE Security Services can authenticate AIX Connections clients. Once authenticated, clients can access DCE and DFS resources. DCE Login is completed under the covers for the PC clients with their normal workstation logons—no additional steps are required.

Key features of the AIX Connections DCE Integration are as follows:

- Supports PC clients (Windows® 95, Windows NT™, Windows for Workgroups (WFW), and OS/2) with the client software shipped with the PCs
- Uses DCE Security server for client authentication; provides a single point of security control and avoids the problem of supporting passwords on every AIX Connections server
- Provides access to DCE resources with full Access Control List (ACL) support

The following startup steps are necessary for DCE Integration:

1. Install and configure DCE and DFS client components. You will need—at a minimum—the following software and any prerequisites for these components: DCE Security Services Client, DCE Cell Directory Services (CDS) Client, and DFS Client (if DFS access is necessary).
2. Run DCE Integration Setup Script `acsetup.dcecp`, which can be customized for the DCE administration policies of your organization. The script `acunsetup.dcecp` can undo the setup. These scripts are located in the directory `/usr/tn`.

3. Enable users to access DCE Integration features. Run the `tndcepasswd` command for each user who will be accessing DCE. It updates DCE Security registry with AIX Connections-specific information for the AIX Connections user or DCE principal. The current release also requires the DCE principal to be an AIX user. This command can be used to change the user's DCE and AIX Connections passwords.

4. Define an AIX Connections Service for DCE access. This service must have the command attribute as `/usr/tn/smb/DCE_LM` file. It can be defined through SMIT or the Web-based administration tool.

5. Start AIX Connections DCE Support. You must be DCE authenticated as `aconnadm` to start the service. Use SMIT or the AIX Connections `tnstart` command to start the NetBIOS realm.

HACMP Support

AIX Connections works with a variety of HACMP configurations to provide highly available services. AIX Connections supports the following configurations:

- Hot standby
- One-sided takeover
- Mutual takeover

These configurations must be set up to meet the following criteria:

- The `/usr/tn` directory and the home directories of AIX Connections users are located on the shared volumes, and symbolically linked from their normal locations.

- The system authorization files (`/etc/passwd`, `/etc/group`, `/etc/security/passwd`, and `/etc/security/group`) are kept in sync across the configuration.

- AIX Connections start and stop commands are specified to HACMP.

AIX Connections does not support HACMP concurrent access mode.

AIX Connections Release 2 provides interoperability with existing AIX features, such as HACMP, plus enhanced system administration and configuration capabilities.

Forward (Proxy) Authentication

Release 2 includes the Forward (Proxy) Authentication feature in the NB and NW realms. This feature enhances interoperability with Windows NT, OS/2 Domains, and Novell® NetWare servers.

When proxy authentication is enabled for a file service, all login requests are forwarded to the file server that is configured as the Proxy. You can configure this feature from the AIX Connections SMIT panels or the Web-based administration tool. When you create a new file service, complete the authentication proxy field by entering a service name, including the file type. The authentication proxy can be the name of a Windows NT, OS/2, NetWare, or AIX Connections file server.

Once proxy authentication is enabled, the AIX Connections service that is enabled for the feature forwards the authentication request to the configured Proxy server. The Proxy server maintains a list of authenticated users and passwords for all file services in its realm. Although this feature allows PC clients to be authenticated by a Proxy server, it still requires the PC clients to have an AIX user account.

OS/2 Extended Attribute Support

Release 2 supports OS/2 Extended Attributes, that is, High Performance File System (HPFS) Extended Attributes. An example of an extended attribute is a file type that indicates whether the file is a command, a text

file, or an executable. This feature allows many PC applications, such as the OS/2 Presentation Manager® drive browser, to function correctly. Without extended attribute support, this browser cannot display file names correctly.

AIX Connections FDDI Support

Release 2 now includes Fiber-optic Data Distribution Interface (FDDI) LAN support, which enables 100 Mbps communications connectivity support to AIX Connections servers. These new product features support FDDI:

- RFC (TCP/IP) NetBIOS support over FDDI
- Additional FDDI IPX/SPX support; FDDI_SNAP and FDDI_802.2 frame types now supported
- Enhanced SMIT supports FDDI capabilities and auto detection of IPX network numbers and frame types

MAC Token-Ring Support

AIX Connections Release 1 ran strictly across the Ethernet™ interface—the IEEE® 802.3 standard. Release 2 supports the Token-Ring interface in addition to Ethernet. Note that the AT server can only be run over one interface in the system. If more than one interface is configured, the server and its daemons will not start.

The AT server interface can be configured two ways. The simplest is to use the Quick Start option from SMIT, which allows you to easily configure the server. Other options include configuring the interface via SMIT, the command line, or the Web-based administration tool.

Web-based Administration Tool

The Web-based graphical user interface allows administrators to change server parameters dynamically and manage the server using a standard Web browser—either from the host or from a remote location such as a client. The administration tasks are secure because the tool requires an AIX password login before any configuration or administration activities can be performed.

The Web-based system administration tool automatically starts when the system boots or it can be manually started and stopped from the command line or through SMIT.

Installation. The Web-based tool includes a `httpd` daemon that is normally configured to run on port 7777. The installation process configures the Web-based tool to use a default port number 7777 for access. Users can access the tool from any Web browser by entering http://fully_qualified_hostname:7777 and the Initial Selection menu page will be displayed.

Main Menu. See Figure 3 for the Main menu.

Access. To access the Tools menus, users must enter a valid AIX user ID and password on the initial screen. From the initial screen, users can choose which part of AIX Connections they want to work with: the initial setup, the Main menu, status, or TotalPrint. After viewing the initial selection screen and choosing a Main menu option, you must enter a user ID and password.

Contents. The Main menu contains seven subjects, shown in Figure 3. The General Management and System sections handle the overall system functions and allow

Figure 3. Web-based Administration Main Menu

users to view status, create user IDs, create and manage volumes, printers, and create customer service information requests for problem determination.

The remainder of the Main menu is divided into realm-specific sections for NetBIOS, NetWare, and AppleTalk®. Within each of these sections, users can configure and administer the realm.

The last bullet on the Main menu allows users to get to the Online Publications—Welcome Page.

Help. Many screens and input fields offer help—available by clicking on the "?" button.

Wizards. Wizards step administrators through configuration windows to ensure that all related items are configured. The wizard lists all necessary steps for adding and/or configuring an item, such as the initial AIX Connections configuration after installation.

Other wizards are available for configuring NetBIOS, IPX/SPX, and AT transports. These wizards follow lists that configure transports and their associated interfaces, routes, and so on. This ensures that all necessary items for the transport to function are configured simultaneously.

Templates. Templates are files that contain commonly used function attribute sets. For example, a user can create a template to add a shared-mode service. The attributes of the specific service are saved in the template. Once a template is created, menu items allow service, volume, and printer creation to create the object using a template. If the template is correct, the system administrator's tasks of creating objects becomes easier and consistent.

Status at a Glance. The Status at a Glance page provides a detailed status of each realm, indicating whether the overall system is up. It also lists the configured services, displaying the status of each service plus the number of clients that are connected. This page is automatically updated every 30 seconds to reflect the current status of the AIX Connections system. See Figure 4.

Customer Service Request Information. The shell script `csr.tn` helps determine configuration problems in AIX

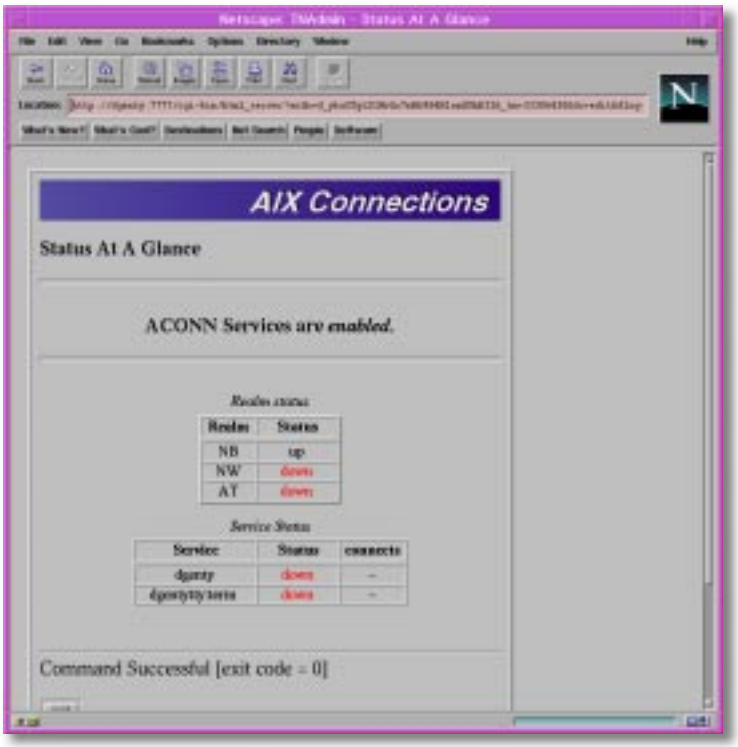


Figure 4. Status at a Glance

Connections. It records configuration parameters, log information, and connection statistics. This information is written to `stdout` and contains output from the `netstat` command, `ifconfig`, `tninfo`, `tnck`, `tnstat`, environments. It also displays the configuration files `/usr/tn/config.tn`, `profile.file`; the services files; the software listing; and any log file information. This information can also be used for debugging purposes if it is redirected to a file.

Online Publications. Online Publications provides a list of all current documents, such as Quick Beginnings and the Administration Guide—a useful feature of this Web-based tool. Located at the bottom of the Main menu is an entry "Online AIX Connections Documentation." Click on this entry to see the publication's "Welcome Page," shown in Figure 5.

You can also view the latest AIX Connections README file. Click on one of the documents, such as Quick Beginnings, which displays a table of contents with links to specific chapters and sections for easy retrieval of information.

From the bottom of each page, you can navigate to the top of the document, to the next document in the list on the Welcome Page, back to the Welcome Page, or to the glossary, which provides links to specific AIX Connections commands. You can also quickly view the appendices of each document.

New Commands and Utilities

New commands that operate on all realms or a specific realm now replace many AIX Connections Release 1 commands. You can now perform administration tasks from the command line or from the Web-based tool. The AIX Connections Reference Guide has a complete listing of the commands. Figure 6 shows some of the new commands.

SMIT Enhancements

Many changes that occurred in the configuration of the AIX Connections servers caused us to modify the SMIT configuration

Figure 5. Online Publications—Welcome Page

New Commands in AIX Connections Release 2

Administration

`tnstart`	Start the service processes and allow client connections
`tnshut`	Performs an orderly shut down of AIX Connections; can specify the number of minutes until disconnect
`tnaccept`	Changes the state of selected services to accept client connections
`tnistat`	Displays the status of the IPX/SPX network interfaces; reports the status of the `tnipx` protocol streams module; shows the IPX routing information
`tnkill`	Disconnects one or more service clients
`tnreject`	Rejects attempts by new clients to connect to file services

Configuration

`tnservice`	Add, modify, or delete file and non-file (terminal) services. Services are defined by providing service names to clients so they can locate and acquire access to share resources on the server
`tnvolume`	Add, modify, or delete volume configurations
`tnvref`	Add, modify, or delete volume references defined for file services. A volume reference must be created for a volume to be accessible in a realm since volumes are created at the system level
`tntransport`	Add, modify, or delete transport configuration options. Transports are low-level networking protocols defined at the system level and referenced from the support realms by using the `tntransport` command
`tniface`	Add, modify, or delete transports for networking interfaces; transports contain a set of network interfaces over which they operate
`ipxprobe`	Displays the frame types and network numbers that are active on an interface. This information is used when configuring the interfaces in the NW realm

Figure 6. New commands in AIX Connections Release 2

process. We also incorporated two new menu options from the main AIX Connections menu: the Quick Start and the Client options. The Quick Start option creates a basic configuration that enables the servers to come up and begin serving clients quickly. The Client option makes it easier to use the client part of the AIX Connections package.

How the volume for a file service is managed shows the differences between the two SMIT versions. In Release 1 of AIX Connections, you first choose the server, then the part of the configuration that you want to work on. For example, if you wanted to configure a Macserver volume, you would first go to `smit aconn->MACserver->File and Print Services-> Configuration-> volume`.

This is a two-step process in AIX Connections Release 2. You do a `smit aconn-> Configuration->Volumes-> Manage Volumes->Volume Name` to create the volume (see Figure 7). You then go through `smit aconn-> Configuration->Manage Volume References` to make the Volume accessible by the realm you want to use (see Figure 8). Once the volume is created, you only need to reference it by the different realms that need to use it.

The advantage of Release 2 is that the administration and configuration is very similar for all the different realms. The configuration screens step you through the process much better, which makes it easier to maintain the servers.

Quick Start

The Quick Start option on the AIX Connections Main menu is a new SMIT option available for configuring and starting up the three server realms. This option was designed to allow end users to quickly create a basic server configuration.

Users choose the interface, services specific to a realm, volumes, volume references, and printer references for the AIX printers that are already defined to create the configuration. Once the configuration is created, the Quick Start option starts the daemons for the server, starts the services, and sets up the services to allow connections from clients. User passwords and accounts must be configured by the `tnpasswd utility`, SMIT, or the Web-based tool before these services can be accessed.

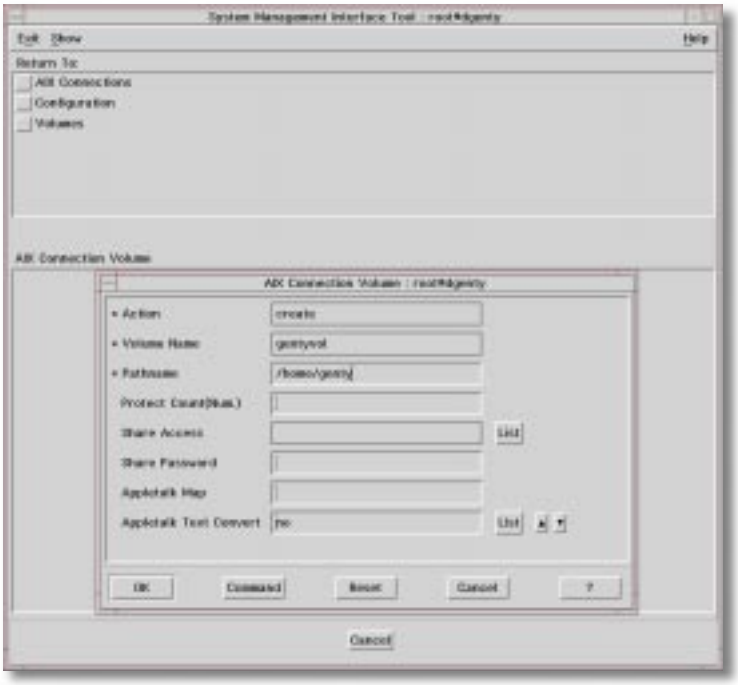


Figure 7. Volume configuration

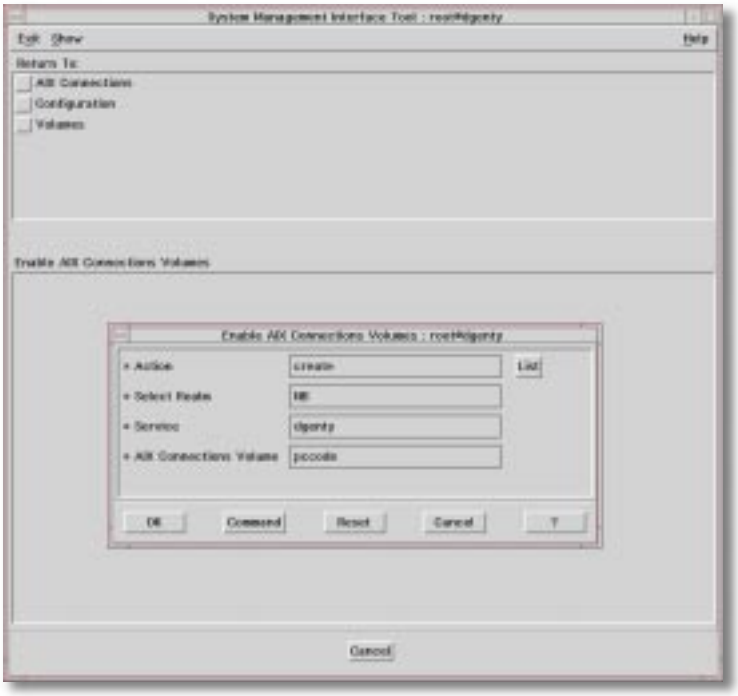


Figure 8. Volume reference configuration

NetBIOS Configuration and Administration

The existing NetBIOS for AIX Administration utility (`mcsadm`) operates as a shell for NetBIOS for AIX software components. It allows users to create and configure LAN Adapters (LANAs), create Resource Control (RC) scripts, manage error log files, start/stop the NetBIOS daemons, and display the status of NetBIOS for AIX.

Users currently enter `mcsadm` on the UNIX command line to invoke the utility, which is a series of panels that allows users to configure or display status or configuration. A NetBIOS command-line interface is available for quick usage. For example, a user may enter `nbix status` to view the status of all configured LANAs instead of viewing them within the `mcsadm` utility. Most of the existing `nbix` options perform the same functions as the screen interface of `mcsadm`.

A new SMIT interface manages NetBIOS for AIX in order to provide a consistent system management interface for AIX Connections and NetBIOS. The SMIT NetBIOS screens allow users to create and manage LANAs, view status, obtain trace files, and manage the WINS server.

The NB realm needs LANAs in order to work. Two types of LANAs can be configured—RFC (TCP/IP) or NetBEUI—depending on the type of protocol needed by the user. Using the SMIT NetBIOS "Add LANA Configuration Option", users input the type of LANA to create, then SMIT automatically creates the necessary Transport Provider Interface (TPI) configuration to run the LANA.

From an AIX Connections view, NetBIOS will be configured with LANAs to get the NB realm running with automatic configuration through the Quick Start option.

NetBIOS WINS Server

The WINS database protocol registers and queries NetBIOS computer names to IP address mappings in a routed network. Using WINS avoids IP broadcasts for NetBIOS names resolution. WINS is well-suited for large, routed IP networks because it reduces the use of local broadcast messages (which reduces network traffic) for name resolution and allows applications to easily locate systems on remote networks. WINS is specially critical in networks that include WAN links. The WINS server can run on the same AIX system as AIX NetBIOS or on a separate system.

> ***The Quick Start option on the AIX Connections Main menu is a new SMIT option available for configuring and starting up the three server realms.***

The WINS server software, configured using SMIT, is automatically installed with the NetBIOS software. WINS configuration has two components: the server on AIX and configuration of client software. The WINS server receives NetBIOS name queries and registrations, which are saved in a database. The clients have WINS software enablement and must be configured to register their name with the WINS server.

Client configuration usually involves inputting the WINS server IP address on the client. Operating systems, such as Microsoft's Windows 95 or Windows NT, can use the AIX WINS server directly. The client registers its name and IP address to the WINS server; the WINS server adds the name and address to the database.

The server ensures that the name is not already used. If a name already exists, the server verifies whether the name is still being used. Once a name is registered, the application can query the WINS server, for example, to identify the IP address of the machine to send data to when a client wants to build a packet. The client software can then send a directed message (as opposed to a broadcast message) to the machine.

WINS SMIT Interface

The following sections describe the configuration of the WINS SMIT interface.

Configuration Updates to mcs0

To configure the NetBIOS WINS server, you must determine the IP interface (or all interfaces) over which the server will run. Advanced configuration options can alter the way in which the WINS server software operates, such as Hash Table size and number of threads. Enter `smit netbwins` to get to the WINS Main menu.

Once the data has been entered on the SMIT server configuration screen, the configuration information is saved in `/etc/mcs0`. A new option was added to the `/etc/mcs0` script—`start_wins`. This option allows users to start the WINS server manually from the command line instead of using the SMIT interface.

When the server is configured through SMIT, you can configure the WINS server to start up at boot time. This adds an entry in `/etc/inittab` after the NetBIOS modules are loaded.

You can start and stop the WINS server and view the status from the SMIT menu. Figure 9 shows the WINS Server Configuration menu.

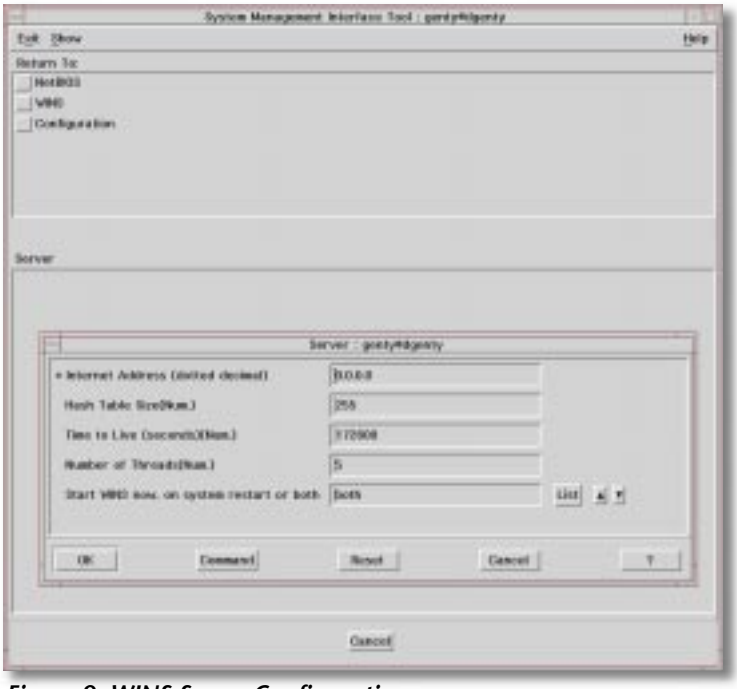


Figure 9. WINS Server Configuration menu

Names

The WINS server maintains two types of names: static and dynamic. For static names, you can enter the machine names in the WINS server table that should have hard-coded IP addresses. These names remain in the table until they are removed through SMIT; they do not require a refresh from the client. There are four different types of static names: unique, group, internet_group, and multihomed.

Clients that are configured to use the WINS server can add dynamic names to the WINS table. The names are added to the table when the client registers, generally at boot time.

The current names in a database can be saved to a file that users can edit. The "Backup Names to file" SMIT option saves the names to an ASCII file. This is useful to save the names for restoration after boot. A SMIT option, "Restore Names from file", restores the names.

By listing entries in the names database, users can determine if the WINS server is working correctly. If clients are configured to use the WINS server (and have been booted since the WINS configuration), the client's machine name will be displayed from the SMIT path: `NetBIOS -> WINS -> Configuration -> Names -> List of Defined Names`. Each name in the database should have an entry similar to the following, which shows the name of the client, the type of name, and its IP address:

```
ascii—ACONN_NTW
hex—41 43 4F 4E 5F 4E 54 57 20 20 20 20
20 20 03
Type: Unique; State: Dynamic; Index: 21
IP Address: 9.3.149.253; Age: 0
```

TotalPrint

TotalPrint provides the features not commonly found in standard AIX print spoolers. PC clients of the AIX Connections servers can submit jobs to the TotalPrint print queue, and then modify those jobs by accessing the queue from a standard Web browser.

TotalPrint includes all standard spooler features such as `lp` or `lpr` with as many features found in mainframe spoolers. These include the following:

- Unlimited number of printers
- Alignment pages
- Job restart
- Job priorities
- Job queue control
- Printer state control
- Automatic user-transparent post-processing of output
- Comprehensive security and accounting features
- Customization and native language support

TotalPrint has comprehensive network support including transparent sharing of jobs and printers across TCP/IP networks, and transparent sharing of jobs and printers using MS-DOS/Windows workstations.

AIX Connections servers can handle print jobs from clients using an AIX print spooler such as `lp`. Depending on the AIX Connections configuration, the TotalPrint spooler can do one of the following:

- Spool jobs to a queue where they will immediately be sent to the print device. The options for the spool command are set in the service-specific configuration file.

- Spool jobs to a queue where they will remain until the queue is notified to print the file, via a separate AIX application that also has the ability to modify the print job's attribute.

A Web browser using command-line applications on its back-end controls the AIX applications that manage the printers and print queues. The client prints jobs as it normally does—by submitting jobs to a network printer. If the job has been spooled to a queue requiring notification to print (as described in the second bullet above), the client must interactively send the job to be printed, using a Web browser, after the job has been "printed" by the local application. This allows the client to modify the attributes of the job before it is sent to the print device.

Security

Access to all print objects (printers, queues, and jobs) is determined according to the AIX user and group of the person attempting to access them. The print objects are represented by AIX files; regular AIX file permissions are used to control access.

SMIT vs. Web-Tool vs. Command-line Configuration

SMIT, the Web-based administration tool, and the command line provide a way to configure the AIX Connections servers. The following example demonstrates some differences between the three methods.

A sample configuration of the NB realm server using SMIT Quick Start follows:

- At an AIX command prompt, type `smit ezaconn`.
- Select NB.
- Select the NetBIOS protocol (for TCP/IP, select RFC; otherwise, select NetBEUI).
- Select the device interface for running NetBIOS.
- Specify Service Name; the default is your AIX System hostname.
- Press Enter.

These steps will configure a file server and a terminal server. It will create the AIX Connections interface, create and start a NetBIOS LANA running over the selected interface, and start the server. Note that the Quick Start option should be used only for an initial configuration. If it is used to reconfigure a server, the results are unpredictable. Use the SMIT Configuration options to further configure or modify your servers.

A sample configuration from the Web-based tool would be similar to the following:

- Create and start the NetBIOS LANA's for the server interface. Use SMIT NetBIOS to do this.
- Connect to the http server through your favorite browser.
- Select Initial AIX Connections Configuration.
- Log in as root.
- Select AIX Connections Initial Configuration.
- Select Configure Administration Information. AIX Connections is required to be the admin user and group.
- Select Update Enable Alternate NetBIOS and specify your server name (your AIX System hostname is a good choice).
- Select the interface that will have to be configured using the NetBIOS configuration procedures before the server can be started.
- Specify your domain.
- Select Update NetBIOS Configuration.
- Go to the Main menu.
- Select File Services and choose your file service.
- Select Volume References and create your volume reference.
- Specify the volume you want the reference to and submit.
- Go back (select back arrow) to the Alternate NetBIOS/NetBEUI Service screen.
- Select Printer Reference and create a printer reference.
- Select Start Service.

The file service that you created should now be accessible.

```
/use/tn/tnstype -A -r NB -t file -a type=0x20 \
            -a description="IBM NetBIOS file service"

/usr/tn/tnstype -A -r NB -t term -a type=0x20 -a \
            description="IBM terminal service NetBIOS"

/usr/tn/tntransport -A -n altnb -a template-only=off

/usr/tn/tnservice -A -r NB -s <server name>:file -a \
           command=/usr/tn/NB/LMfile -a transport=altnb \
           -a persistent=off -a client-encryption=on -a \
           description="NB server on <server name>"

/usr/tn/tnservice -A -r NB -s <server name>tty:term -a \
            description="NB term on <server name>" -a \
            command="/usr/tn/NB/NBtty /usr/bin/login"
```

Figure 10. SMIT NetBIOS commands

For the same configuration that you would get from the SMIT NB realm Quick Start, follow these steps on the command line:

- Create and start the NetBIOS LANAs for the interface that you want to run the server over by using SMIT NetBIOS.
- Run the commands shown in Figure 10.
- Run the following for all of your AIX printers:

```
/usr/tn/tnpref -A -r NB -s <server name>:file -p ${i}
```

where `${i}` is an AIX printer.

- Create volume references for pccode and home volumes, as shown in Figure 11.
- Create your interface.

```
/usr/tn/tniface -A -i <interface> -n altnb \
                  -a device=/dev/lana<number>
```

- Start your server.

```
/usr/tn/tnstart -r NB
```

```
/usr/tn/tnvref -A -r NB -s <server name>:file  -v home
/usr/tn/tnvref -A -r NB -s <server name>:file -v pccode
```

Figure 11. Volume references for pccode and home volumes

Similar configurations can be created for the NW and AT realms. Additionally, you can configure everything separately from both the SMIT configuration and the Web-based administration tool.

***Denise Genty**, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Ms. Genty is a staff software engineer in the RS/6000 Division currently working on AIX Connections and NetWare for AIX. Her primary areas of concentration are AIX device system configuration and communications applications. Ms. Genty has a BS in Computer Science from Texas A&M University.*

***Michael S. Lew**, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Lew is a software engineer on the PC Interconnectivity team in the RS/6000 Division. His area of focus is hardware diagnostics, communications, and AIX software development. He has a BS in Electrical Engineering from Texas A&M University.*

***Rakesh Sharma**, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Sharma is an advisory programmer with a specialty in networking protocols, TCP/IP, and UNIX interoperability with PCs. He is currently working with NT/PC interoperability for AIX. He has a BS in Electrical Engineering from ITT Kanpur in India and an MS in Computer Science from North Dakota State University.*

AIX Connections: Seamless PC Security

By Kay Chang and Rakesh Sharma

AIX Connections Release 1 supports PC LAN protocols in a workgroup environment. Its system software enables networked PCs running popular PC operating systems (such as Windows 95, OS/2®, Windows NT™, and Macintosh) to operate from a single server—the RISC System/6000® (RS/6000™). An additional new function provides a secure directory in which PC users, who access various domains and protocols, are authenticated to the enterprise system via DCE security. PC workgroup users now use the DCE domain for accessing DFS files, as well as AIX Server Journaled File System (JFS) files, even within the Web browser.

The goal of AIX Connections is a single system that controls and manages multiple networked PCs without requiring any modification of PC clients. The previous release of AIX Connections integrated diverse PC desktops onto a single AIX server. However, when a single AIX Connections server reaches its limit of PC clients (limit is based on the server's resource capacity), AIX Connections must merge multiple AIX Connections servers without interrupting PC clients, yet present a single point for administration and access for all PCs within its domain.[1]

Distributed Computing Environment (DCE) enables AIX Connections to provide high-level, integrated services for consolidating multiple AIX Connections servers.

AIX DCE provides the following functions:

- DCE security with full encryption and authentication
- Global file sharing with Distributed File Systems (DFS)
- Global Directory Services to integrate users and user groups
- DCE replication function that improves reliability of server applications

With AIX Connections, users can share files, printers, and even administration across different types of networked PCs. With enhanced DCE support, AIX Connections now incorporates DCE functions into the standard PC clients. AIX already provides a base AIX function to DCE clients. Enhancing AIX Connections with DCE provides an efficient environment for all PC clients that are gateways to the DCE domain and its services.

In AIX Connections Release 1, three well-known file and print servers (SMB, NetWare®, and Apple®) using their respective PC protocols (NetBIOS, IPX/SPX, AppleTalk®) matched access and permission to the AIX users and group permission. The new release with DCE enables AIX Connections to improve this capability by extending it to

Kay Chang

Rakesh Sharma

[1] Olszewski, Bret, R. and Chang, Kay. "AIX Connections." *AIXpert*, November, 1995. Pages 4-13.

the DCE domain, providing file extensibility and security. It uses DFS to provide distributed files and it also provides a single point of control for many servers.

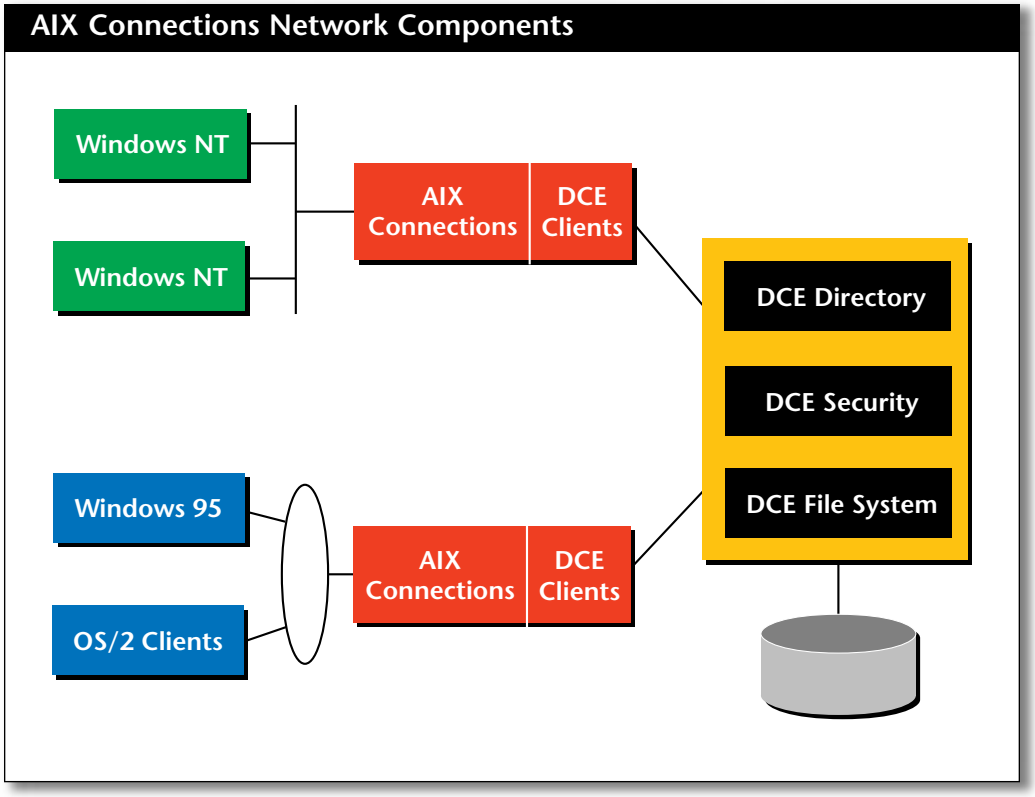


Figure 1. PCs, AIX Connections servers, and DCE components in networks

AIX Connections—DCE for PCs

The following sections describe the details of the new AIX Connections with DCE capabilities.

PC Clients

AIX Connections requires no changes for existing PC clients. It supports PC Server Message Block (SMB) clients, in which each user password is intercepted by AIX Connections and authenticated by the DCE Security service.

AIX Connections uses DCE `Cell_Admin` services to set up PC users and groups before the PC clients log in. The DCE-provided Extended Registry attribute schema also needs to be set up. A PC client accesses network resources by "mapping a drive," which triggers the AIX Connections server to set up a session. AIX Connections authenticates the user and user groups for the DCE realm, which allows the requested PC clients to access required resources, such as shared applications or data files, or shared printers. AIX Connections can be used to change PC client passwords within the DCE security environment.

Web DFS Support via SMB-Enabled Browser

Microsoft Internet Explorer supports the SMB-based file Common Internet File System (CIFS) in Windows 95 and Windows NT. The result is AIX Connections having DFS file support within the browser. This is important because it uses a file protocol rather than a HTTP Web server protocol to access and update Web pages. This browser can also access pages in a server that is not a Web site.

For example, in Internet Explorer, the URL may look like the following: file://SMBSERVER.austin.ibm.com/sharedDFSVolume/filepath. The SMB server is integrated into the AIX Connections SMB support, then mapped into the DFS file space. This function can be done without modifying a single line of the PC client's function. This feature requires a CIFS-supported browser.

AIX Connections—DCE Components

The `DCE_LM` file process performs the PC client login validation. The DCE administrative cell opens the administrative process. Figure 1 shows the integration of PCs, AIX Connections, and DCE.

Networked Environments

Networked PC environments, particularly Microsoft networks and the IBM LAN Server domain, use NetBIOS and SMB protocols for file access and system administration. The most common protocols are NetBEUI, NetBIOS over TCP/IP, and various levels of LAN Manager protocols. Today, AIX Connections handles NetBIOS protocols and some SMBs. Although new, extended SMBs, especially those that deal with OS/2 LAN Server Domain Controller and Windows NT Trusted Domain are not yet implemented, AIX Connections proxy services can provide interoperability with OS/2, Windows NT, and Domain Controllers.

AIX Connections already supports underlying network protocols, such as NetBEUI, NetBIOS over TCP/IP, and IPX/SPX2. It also supports some service protocols, such as file and print SMBs and NetWare Core Protocols (NCPs). AIX Connections also supports additional features, which are now required, such as user login validation and user and group permission rights in a global enterprise.

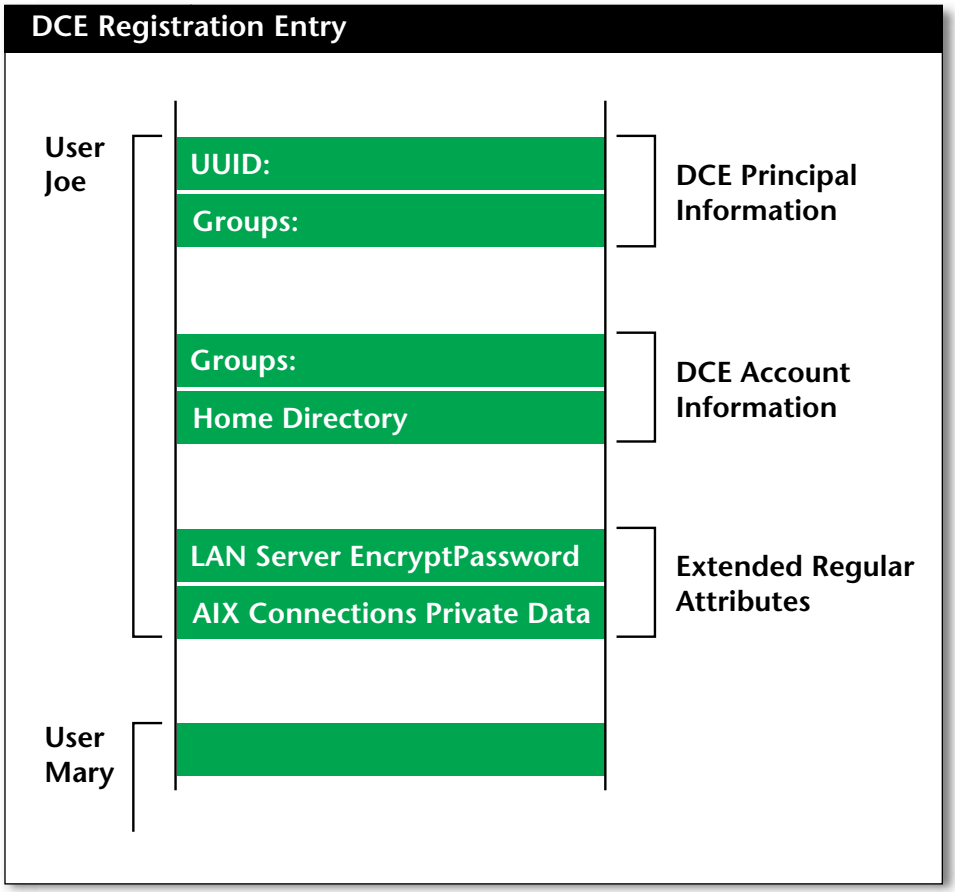


Figure 2. DCE registration entry for AIX Connections-based PC users

DCE Registration of PC Users

The AIX Connections administrator registers PC users to the DCE registry by using the Extended Registry function. Using the acesetup.dcecp script, AIX Connections administration principal, groups, and Extended Registry attributes objects are created for a secure DCE environment. Once the AIX Connections principal is created, the DCE System Management Interface Tool (SMIT) can set up users who will participate in a DCE domain. Both establishing initial passwords and changing user passwords are done using `tndcepasswd` by either the `cell_admin` or by a DCE user who is already logged in.

Figure 2 shows an example of a user's DCE registry entries.

> *These new functions of AIX Connections bring the robustness of services that have never existed for PC clients.*

PC User Session Setup

Figure 3 shows the user authentication process for PC clients during a session setup request. The flows between the clients and AIX Connections LAN Manager server are well-known, non-modified SMB flows

between SMB clients and servers. The AIX Connections server uses the DCE Extended Registry to retrieve the LAN Server encrypted password and the "clear" password to authenticate a PC user. The LAN Server's encrypted password verifies the PC user password, and the "clear" password is used for DCE login in the DCE cell domain.

Putting It All Together

Figure 4 shows the actual components and their interactions when a PC client tries to access a DFS file. The scenario includes DCE user authentication and file access permission grants. Note that step in `tndcepasswrd`. The DCE Security server is invoked when a password must be changed.

The DCE enhancement positions AIX Connections to provide services for non-modified Windows clients.

Conclusion

The DCE enhancement for AIX Connections is a first step for including PC users for Windows 95 and Windows NT clients that participate in the enterprise directory and security. This enhancement positions AIX Connections to provide services, such as DCE directory and security for non-modified Windows clients. It also enables Windows

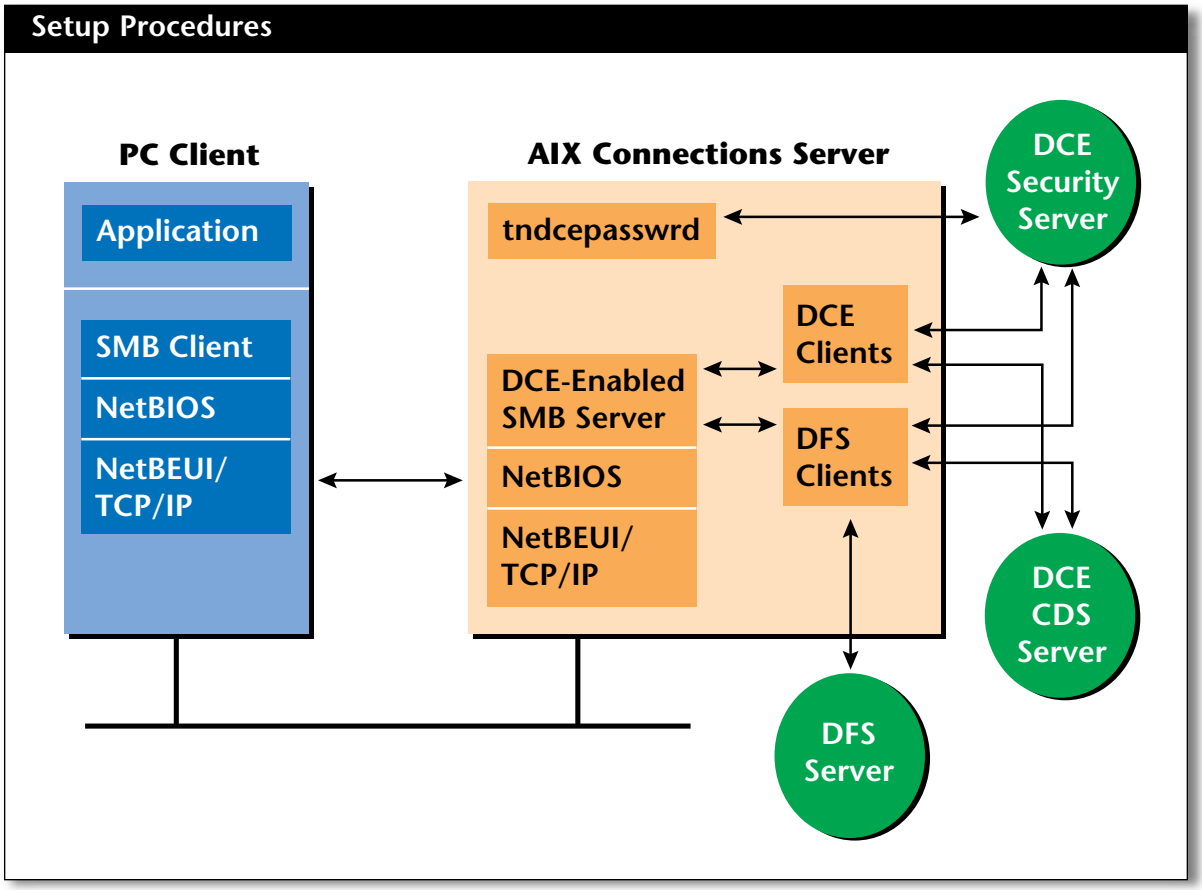


Figure 3. Session setup procedures between PCs, AIX Connections, and DCE Security server

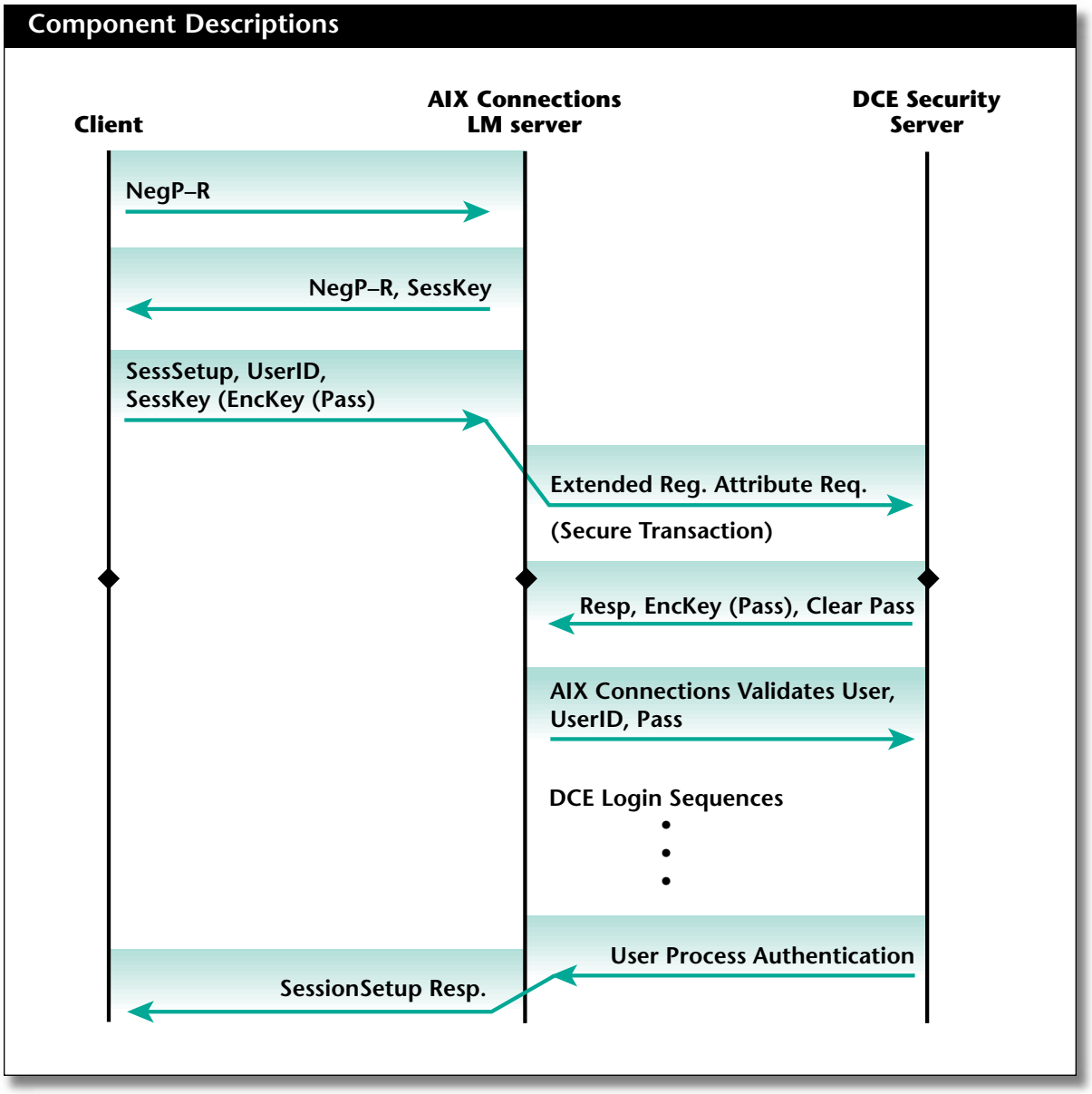


Figure 4. PC client accessing DFS file

clients to access the DCE file systems, which are secure, reliable, and distributed. These new functions of AIX Connections bring the robustness of services that have never existed for PC clients.

***Kay Chang**, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Ms. Chang is a senior programmer in AIX communication architecture working on AIX Connections and the Network Computer. She has an MS in Computer Science from Wright State University in Dayton, Ohio.*

***Rakesh Sharma**, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Sharma is an advisory programmer with a specialty in networking protocols, TCP/IP, and UNIX interoperability with PCs. He is currently working with NT/PC interoperability for AIX. He has a BS in Electrical Engineering from ITT Kanpur in India and an MS in Computer Science from North Dakota State University.*

The IBM Network Station for AIX

By George Kraft IV

The IBM Network Station is a true network computer device. It commands the power of the Internet while reducing the total cost of ownership.

Today's Internet is the result of heterogeneous computing environments embracing and cultivating open systems standards for network computing. As the demand for network computing continually increases, many new technologies and potential standards will emerge.

The Internet was the impetus for the Network Computer Reference Profile (NCRP) initiative, introduced by Apple®, IBM, Netscape®, Oracle®, and Sun®. Its purpose is to facilitate a broad application base, interoperability among systems, simple and unified system administration, and end-user ease of use for a network computer device. The device itself should be architecturally neutral and secure; it should also have a low total cost of ownership.

The NCRP addresses the various markets for the network computer including the consumer, education, government, developer, manufacturer, and corporate markets. Specifically, it outlines the technologies and standards shown in Figure 1.

IBM Network Station

IBM's initial response to the NCRP is the IBM Network Station™. Architecturally, the IBM Network Station is server independent. It is deployed for various IBM platforms

NCRP Technologies and Standards

Resources
- Minimum 640 x 480 resolution (or equivalent)
- Pointing capability
- Text input capability
- Audio output
- Persistent local storage not required

Internet Protocol
- Internet Protocol (IP) Version 4
- Transmission Control Protocol (TCP)
- File Transfer Protocol (FTP)
- User Datagram Protocol (UDP)
- Domain Name Service (DNS)

World Wide Web
- HyperText Markup Language (HTML) Version 3.2
- HyperText Transfer Protocol (HTTP) Version 1.0
- Java™ Application Environment Version 1.0.2

Electronic Mail
- Simple Mail Transfer Protocol (SMTP)

Multimedia
- JPEG
- GIF
- WAV
- AU

Security
- Secure Sockets Layer (SSL) Version 2

Figure 1. Technologies and standards addressed by NCRP

George Kraft IV

Figure 2. The IBM Network Station off-loads work from the server distributively

running AIX®, OS/400®, OS/390™, and Microsoft® Windows NT™.

The IBM Network Station Models 8361-100 and 8361-200 provide Ethernet™ and Token-Ring access, respectively. The unit has a 33 MHz PowerPC™ processor, 1 MB of video RAM expandable to 2 MB, 8 MB of RAM expandable to 64 MB, AC/DC converter, mouse, and keyboard. The monitor is not included; however, PC monitors such as the IBM G-series and P-series are an optimal price-versus-performance choice.

The IBM Network Station has X-Terminal-like traits but with the advantage of running client applications locally, such as terminal emulators, the Java Virtual Machine, and World Wide Web browsers. The ability to run these tasks locally on the IBM Network Station is important because users receive better response time and it off-loads work from the server distributively.

In addition to displaying distributive X Window System® applications, the IBM Network Station can run server-resident Microsoft Windows™ applications by using WinCenter™ Pro from Network Computing Devices®, Inc. Native server applications run on the host, then display to the network computer device.

Terminal Emulators

Telnet, 5250, and 3270 terminal emulations support character-based input for UNIX®, AS/400®, and OS/390 respectively. Each emulator runs as a local application on the network computer device, which allows the end user to access multiple server applications simultaneously from a single IBM Network Station.

Java Virtual Machine

The Network Station also provides Java Virtual Machine (JVM) Version 1.0.2 and Java class libraries. The JVM runs Java applications and applets locally on the network computer device. The write-once-and-deploy-everywhere Java paradigm is an obvious advantage over writing for specific operating systems.

Web Browser

Later in 1997, the Network Station Browser (NSB) and Navio™ Navigator World Wide Web browsers will be available as local clients. Then, users will be able to surf the Web directly from the network computer device.

In AIX 4.2.1, Common Desktop Environment (CDE) users who are using the IBM Network Station now have new Netscape Navigator™ integration that has been added for URL data types in the CDE. Telnet, file, ftp, mailto, news, http, gopher, nntp, wais, and prospero are the new URL data types. URLs can be opened, printed, dragged and dropped, or added to the browser's bookmarks from the CDE File Manager, Mailer, Front Panel, Printer, and Workspace.

AIX Offering

Although the IBM Network Station is architecturally neutral, AIX adds value through its integrated installation and configuration.

Installation

For AIX 4.1.5 and AIX 4.2.1, the installation packages contain the IBM Network Station software: the base kernel and configuration, terminal emulators, JVM, and X-Server. The

software is available via a Program Temporary Fix (PTF) APAR 1X64800 for AIX 4.1.5, with the Base Operating System (BOS) for AIX 4.2.1, or via the World Wide Web for both. See "Surf's Up..." below. World Wide Web browsers will be available in the future.

AIX Services

To expedite the configuration of AIX to support the IBM Network Station, the software installation process configures and enables BOOTP, TFTP, NFS, and the X-Font server. The system administrator simply needs to define some trivial BOOTP device parameters.

SMIT

The installation packages include System Management Interface Tool (SMIT) support for BOOTP devices such as the IBM Network Station. The system administrator runs SMIT with the `bootp fastpath`, where the IBM Network Station devices are easily set up by naming the device, Media Access Control (MAC) and IP addresses, boot server, gateway, and subnet mask.

For specific system administration documentation for the IBM Network Station, see the URL file://usr/netstation/doc/SysAdminGuide.html on an AIX server after installation of the install packages.

X Integration

Users can X Display Manager Control Protocol (XDMCP) authenticate for the X Window System using `xdm` or `dtlogin` as they normally do for AIX. The IBM Network Station can integrate with CDE, the Motif® window manager, or it can run as a stand-alone with its own local window management. For specific user information, see the following URL file://usr/netstation/doc/UsersGuide.html.

The server capacity required to host the IBM Network Stations depends primarily on how much the local clients are used in place of the desktop services found on the server. Each function, such as a Web browser or Java application, that is moved to the client, reduces the load on the server and increases its capacity to host additional network computer devices.

Migration

The IBM Network Station opens a whole new world for AS/400 and S/390® users. They now can have multiple and distributive applications, and the World Wide Web in a Graphical User Interface (GUI) environment. The transition from "green screen" terminals with function-input to windows with menu bars is an evolutionary step forward.

On the other hand, AIX and IBM X-Station customers have been surfing the net from the beginning. The transition to the IBM Network Station is minimal due to support of open system standards and protocols.

The Java Virtual Machine and World Wide Web browser capabilities have made this network computer a new and exciting network computer device. Instead of just viewing applications, users can dynamically load and run Java applications and applets; they also can browse the World Wide Web. These new capabilities make the IBM Network Station a natural three-tier client/server solution.

Surf's Up...

The IBM Network Station total cost savings is real; it connects easily into existing AIX environments; and it bridges into the future of network computing.

For more information on the IBM Network Station, see the World Wide Web at http://www.internet.ibm.com/computers/netstation/ or call 1-800-CALL-IBM. For the latest IBM Network Station software, see http://rchas943.rchland.ibm.com/nc/swdist/.

George Kraft IV, *IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Kraft is an advisory software engineer for IBM's new Network Computer Division. He recently moved from IBM's RS/6000 Division where he worked on the AIX integration of the IBM Network Station. He has a BS in Computer Science and Mathematics from Purdue University.*

Testing Cluster Solutions[1]

By Steve Nasypany, Mike Panico, Sonia Weaver, and Juan Zalles

Our RS/6000 development team installed, configured, and tested a variety of existing products to see how they functioned in a clustered environment. We focused on how these individual products increased the availability, manageability, and scalability of the cluster. Some of the products tested include HACMP, Distributed SMIT, Network Installation Management, Tivoli Management Environment, ADSTAR Distributed Storage Manager, Performance Toolbox, and Lotus Notes.

Today's business-computing environment consists mainly of networked machines. Many businesses are beginning to "cluster" their machines to improve performance, availability, manageability, and scalability. A *cluster* is a group of servers that appear to client machines, administrators, and programmers as a single computing resource.

The move to clustering is already taking place within businesses, and now hardware and software developers must play catch-up to engineer products that take advantage of clustering principles. But you don't have to wait for a cluster solution—you can create one now.

What is a "cluster solution" and exactly what problems are these clusters solving? The answer to this question depends largely upon your specific environment, and what you wish to achieve. For the purposes of our testing, we focused on a RISC System/6000® (RS/6000™) environment using the AIX® operating system. In this environment, three goals were of top priority:

- High availability to isolate or reduce the impact of machine, resource, or device failures
- Manageability to balance loads and reduce system management costs
- Scalability to expand the capacity of servers, clients, users, or other resources

Our cluster development team at the IBM Austin site has taken existing software products running concurrently on a collection of RS/6000 hardware to produce a clustered environment. We performed dozens of test cases to discover how these products achieved the above goals. This article summarizes the results of these tests.

After reading the article, you should have a general understanding of the considerations involved in using a variety of products in a common cluster configuration. You should also come away with information (including suggestions) about specific products that might be useful for your environment, even if

Steve Nasypany

Mike Panico

[1] This article also appears on the Web. Open URL: http://www.rs6000.ibm.com/resource/technology. Select the "High Availability & Clustering" link or scroll down to that section. Select the link entitled "Testing Cluster Solutions in an RS/6000 Environment."

you do not use the exact RS/6000 environment we have tested. This article also includes pointers to product information that you can order or view from the World Wide Web.

Summary of Test Cases and Results

For testing, we selected a wide variety of products that are representative of a typical AIX environment, including system management, maintenance, load balancing, printing, and middleware products (as they became available).

We tested the software products shown in Figure 1 in a clustered environment. For each product, we discuss the test case results as they apply to high availability, manageability, and scalability. While some products serve all three goals, other products are applicable to only one or two of these goals.

A 4x2 cluster scenario (four two-node clusters) was used to most accurately emulate a realistic customer cluster environment. Our test plans called for one of the two-node clusters to represent the "Corporate" home office and the other three two-node clusters to represent remote sites, or "Branches." We planned to manage two of the remote sites with the corporate cluster, leaving one to be self-managed (the "Super Branch") with backup from the Corporate cluster. The test plans also called for us to create and extend the remote clusters.

Figure 2 shows the hardware used in our test cases.

Additionally, we used an IBM 3116 Page Printer with a Token-Ring network interface.

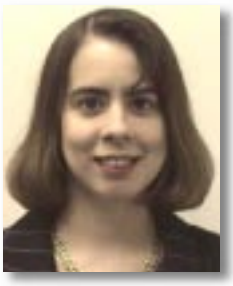
The following sections describe each product we tested, the purpose of each test case, and the results as they apply to high availability, manageability, and scalability.

AIX 4.1.4

We chose the AIX 4.1.4 operating system because this release of AIX was available and supported each of the products we tested. While we are reporting the results of tests for specific hardware and software configurations, these products run on other RS/6000 platforms (such as the RS/6000 SP™) and software (AIX 4.1.5). These test case results should still be useful for planning cluster solutions in your environment.

We installed the base Server package for AIX 4.1.4 on all servers. Roughly three dozen additional operating system filesets were installed, primarily to support the test suite products and device drivers specific to Differential SCSI and Serial Storage Architecture (SSA) drives. We also installed 15 Program Temporary Fixes (PTFs), most of which were specific to new releases of Lotus Notes® (4.11 and 4.5). As of this writing, the fileset updates include:

bos.adt.include	4.1.4.4
bos.adt.prof	4.1.4.17
bos.net.tcp.client	4.1.4.16
bos.rte.libc	4.1.4.18
bos.rte.libs	4.1.4.12
bos.rte.streams	4.1.4.2
bos.rte.tty	4.1.4.13
bos.sysmgt.serv_aid	4.1.4.5
bos.rte.up	4.1.4.17
bos.rte.mp	4.14.17
bos.rte.security	4.1.4.2
bos.rte.libnetsvc	4.1.4.2
bos.rte.printers	4.1.4.7
devices.mca.8ef4	4.1.4.2
devices.mca.ffe1.rte	4.1.4.1



Sonia Weaver

Juan Zalles

Software Products Tested

- AIX 4.1.4
- HACMP/HANFS 4.1.1
- Network Installation Management
- Distributed SMIT 2.2.1
- Tivoli Management Environment 3.0
- Tivoli/Sentry 3.0
- Tivoli/Admin 3.0
- Tivoli/Courier 3.0
- LoadLeveler 1.2.1
- Interactive Session Support 1.2.1
- ADSTAR Distributed Storage Manager 2.1.5.6
- NetTAPE 1.1
- Performance Toolbox 2.1.4
- Print Services Facility 2.1
- Lotus Notes 4.11 and 4.5.

Figure 1. Software products tested in a clustered environment

Hardware Used in Testing

Corporate Cluster

- 2 Model 7013 590 RS/6000s (HACMP servers) with 128 MB of RAM
- 1 Model 7013 530 RS/6000 (HACMP client) with 128 MB of RAM
- 2 SCSI2-DE external hard drives (a 7204 325 with 4.5 GB and a 7204 317 with 2.2 GB)
- 2 Token-Ring adapters for each server node (to allow for HACMP IP address takeover)

Super Branch Cluster 1

- 1 Model 7013 580 RS/6000 (HACMP server) with 128 MB of RAM
- 1 Model 7013 530 RS/6000 (HACMP server) with 128 MB of RAM
- 1 Model 7248 43P PowerPC (HACMP client) with 64 MB of RAM
- 1 7204 325 SCSI2-DE external hard drive with 4.5 GB
- 1 Token-Ring adapter per node

Branch Cluster 2

- 2 Model 7013 530 RS/6000s (HACMP servers) with 48 MB of RAM
- 1 7204 315 SCSI2-DE external hard drive with 2.0 GB
- 1 Token-Ring adapter per node

Branch Cluster 3

- 2 Model 7013 J40 RS/6000s (HACMP servers) with 128 MB of RAM
- 4 Model 010 SSA hard drives with 4.5 GB
- 1 Token-Ring adapter per node
- 1 Ethernet™ adapter per node
- 1 Asynchronous Transfer Mode (ATM) adapter per node
- 1 Fiber-optic Data Distribution Interface (FDDI) adapter per node

Figure 2. Hardware used in testing

HACMP 4.1.1 for AIX (with HANFS)

High Availability Cluster Multiprocessing (HACMP) for AIX provides high availability in a clustered environment. By using shared resources between cluster nodes, it eliminates single points of failure and provides limited downtime in the event of a failure.

For the purposes of this article, the following terms and definitions are used to describe the types of HACMP configurations:

Cascade mode: If a primary node fails, the resources (such as applications, disks, and network address) are acquired by a backup node. When the primary node returns to service, it resumes ownership of its resources.

Mutual takeover: If any node fails, the resources of that node are distributed to other available nodes in the cluster. When the failed node returns to service, it resumes ownership of its resources.

We set up the Corporate cluster for mutual takeover with Internet Protocol Address Takeover (IPAT). IPAT allows another node in the cluster that has a standby network adapter card to assume the IP address of a node that fails. This cluster also had two external SCSI2-DE hard drives, with one node controlling one hard drive and one node controlling the other. If a node failed, the other node acquires control of the external hard drive of the failed node, along with the IP address of the failed node.

We tested the IPAT functions exclusively at the Corporate cluster level. The first two Branch clusters were set up to cascade. In this scenario, a primary node owns a resource (our external SCSI2-DE hard drive). When the primary node fails, the secondary (standby) node acquires the external hard drive. This allows access to any data on that hard drive by logging into the secondary node.

The third Branch cluster consisted of J40s using the PowerPC™ 604 (two-way), along with four SSA external hard drives in a Model 010 drawer. We also configured this cluster for cascading failover. If the primary node fails, the SSA hard drives would be acquired by the secondary node. We used the High Availability Network File System (HANFS) 4.1.1 implementation provided with HACMP instead of using HANFS Version 4.2. The primary reason for using this configuration was because HANFS Version 4.2 cannot operate on nodes running HACMP. Because high availability was an integral part of our clustering test strategy, we used HANFS 4.1.1 combined with HACMP. If your environment requires file locking services, you should implement HANFS 4.2 on a separate cluster without HACMP.

HACMP lets you customize the functionality of the product's operation with scripts that define actions for specific HACMP events. We tried to configure our cluster as generically as possible to reduce the amount of scripting required. Any specific scripting requirements for our test cases are described within the applicable product section.

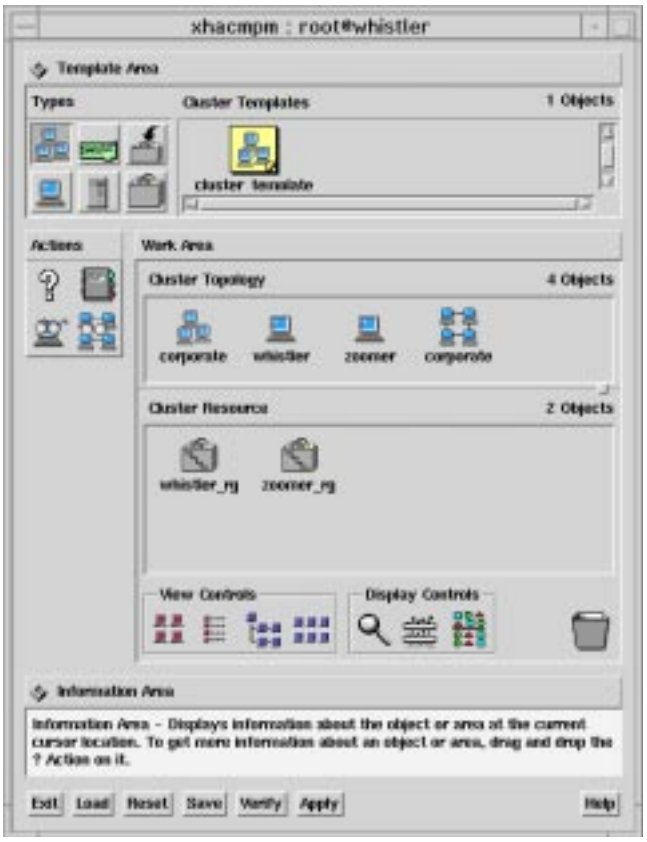


Figure 3. HACMP's Visual System Management interface

High Availability

A suite of regression tests was performed to test each of the major functions of HACMP and HANFS, and to verify our configuration. Other than the problems discussed in the following two sections, the testing went as planned.

Manageability

Each cluster configuration was updated easily using the System Management Interface Tool (SMIT) menus provided with HACMP. We were able to quickly add and modify resources, and the other nodes in the cluster were easily synchronized from the primary node. However, we did encounter one cluster-specific hardware problem during installation and configuration.

This problem involved the SCSI2 Differential Y-cables and device-to-device cables losing contact with one or more of the adapters or drives. This was caused by the weight of the cables, which were not well supported. To avoid this problem, we recommend the following:

- Properly support cables attached to adapters and drives.
- Be careful when moving SCSI2-DE equipped platforms, because movement might loosen the cable or adapter connections.

We encountered another problem in getting HANFS to work properly. The HANFS file systems containing the home directories of all users in the clusters were not available to other components that relied on these directories being available during startup. Until HACMP/HANFS starts, these file systems are not available for mounting because the server node that controls the HANFS file systems can mount these file systems only through HACMP/HANFS.

To avoid this problem, we recommend the following:

- Write post-event HACMP scripts that mount any needed file system upon completion of the HACMP node-up event.
- Carefully plan the startup order of resources on the cluster nodes (particularly in `/etc/inittab`), and adjust for any dependencies required by middleware applications.

Additionally, we encountered a problem with a cluster server node booting on its service address instead of its boot address when the cluster was configured for IPAT. This problem occurred when we manually stopped and started the cluster services repeatedly. To troubleshoot this problem, we checked the network addresses with a command (such as `netstat -i`) and manually restarted the cluster.

Lastly, we encountered a problem importing a volume group from one node to another on the SSA hard drives. To avoid this problem, install the latest updates for the SSA device drivers:

- devices.mca.8f97 and devices.ssa.disk for level 4.1.5.0.

Scalability

When we tested for scalability, we updated one of the three Branch clusters with products to duplicate the Corporate cluster and serve as a "Super Branch." We ran scalability tests in which additional hard drive resources were allocated to the Super Branch cluster. Our testing went as planned.

Network Installation Management

Network Installation Management (NIM) for AIX 4.1.4 installs the Base Operating System (BOS) and optional software on one or more machines in an AIX network environment. A NIM configuration consists of one or more servers providing resources for installations, a set of installable clients, and definitions of the available networks in the environment. An administration server for configuring, controlling, and updating the NIM environment is known as the *NIM Master*. NIM can install to clients and allow clients to download from NIM resource servers.

> ***HACMP lets you customize the functionality of the product's operation with scripts that define actions for specific HACMP events.***

For BOS installations, two primary resources are required. First, the Shared-Product Object Tree (SPOT) provides file system support when a client boots from the network. Second, a source for required software packages used in the installation process must be defined.

High Availability

The NIM tests focused on demonstrating the feasibility of creating and maintaining a highly available resource server and master—one capable of performing BOS and custom fileset installs if a primary NIM master fails. We had to customize the HACMP configuration to create highly available volumes upon which the SPOT, install packages, and `/export` and `/tftpboot` directories could reside.

We began by installing and configuring a NIM master on the primary node of the first cluster. This typically entails defining the network environments and identifying the clients. The HACMP software was started, allowing access to the highly available file systems. The NIM SPOT and install packages were then built on the highly available file systems, along with optional install packages for the AIX server and several test products. This process took from one to two hours, based on the number of optional packages we selected. We then defined optional products to allow them to be installed as custom packages from the NIM master.

Once we had configured the first NIM master, we initiated an HACMP failover using the HACMP SMIT menus. This

allowed the backup node to access the SPOT and licensed product resources defined on the highly available file system. The NIM network, client, SPOT, and install packages were then defined on the backup. This procedure, while repetitive, is not time consuming.

Manageability

The configuration described in the "High Availability" section allows the backup NIM master to access the SPOT, install packages, and `/export` and `/tftpboot` directories if the primary NIM master fails. If a failover occurs during an install, you must enter the identity of the resources and clients being installed to restart the install from the backup.

Because there is no way to synchronize NIM masters, you should carefully plan how you will define resources and clients. For example, the definition of multiple install packages adds to the configuration of the backup master, since each resource definition must be duplicated. If possible, minimize manageability problems by placing optional packages in a single, defined NIM resource. In this case, you would centrally locate installation images in a single directory.

For resource definitions, the highly available file system must be failed over to the backup node. For client definitions, no failover is necessary since the client information is stored locally.

Scalability

The NIM failover capability described in the "High Availability" section does not increase the scalability of NIM by allowing a larger set of machines to be installed. That is because in this setup, only one NIM master is active at a time. For scalability tests, we configured a second cluster identically to the first and performed installs. This capability also provided redundancy if the first cluster failed completely.

A more functional alternative for an environment concerned with maintaining access to non-BOS packages would be to duplicate the packages on several servers. These packages typically require a fraction of the installation time of the BOS packages, and clients can download them from another server if the primary master is busy or is otherwise unavailable.

NIM's overall performance is a function of the resource server, its load, the networks involved, and the number of resources and clients being installed.

Distributed SMIT 2.2.1

Distributed SMIT (DSMIT) lets you build and distribute SMIT commands to other clients on a network. A DSMIT configuration consists of the DSMIT servers, a configuration file server, and its clients. The clients can be grouped into domains, which specify a permanent list of clients to be managed together. During runtime, any set of clients or domains can be managed as a whole. You can define multiple servers, allowing management functions to be distributed across several hosts.

> ***DSMIT lets you build and distribute SMIT commands to other clients on a network.***

For this test case, we focused on making the DSMIT configuration file server a highly available resource, allowing system administration tasks to be performed if the primary node failed.

High Availability

In its current design, the configuration file server holds the DSMIT security configuration files. It is a single point of failure that can hamper high availability. Our test case involved installing a DSMIT server on a clustered pair. To initialize the primary server, we defined it as the configuration file server. We input a list of managing machines that would then have the DSMIT server software installed. Lastly, we defined the administrators and the list of clients on the network to be administered. DSMIT then creates a set of security keys for defining the file server, managing machines, clients, and administrators.

DSMIT was then run to build the lists of clients and servers to be grouped into domains, and to specify which operating system they were using. (DSMIT can support AIX, Sun®, and HP™ clients.) DSMIT can then be started by specifying these domain names. Next, we initialized the backup node and clients using the appropriate keys. The DSMIT file server configuration files were then copied to the backup node.

Upon failover, the backup server assumes the primary master's IP address and can provide the file server functions as needed for a local or remote DSMIT server interface to operate. We successfully performed a series of system administration tasks.

Manageability

DSMIT, like SMIT, provides an easy-to-use interface with comprehensive system management functions. You can use DSMIT to manage one or many clients. For this test case, the primary cluster operated in cascade mode, thus only a single DSMIT server was active at any given time. However, specifying additional hosts as managing platforms allows for multiple DSMIT sessions.

Typically, multiple managing machines are defined to support more than a single administrator. As with all operations in which multiple administrators are performing system management functions, be sure to coordinate the operation of maintenance tasks.

Scalability

While our configuration involved copying identical domain and client information to each DSMIT server, this is not a requirement. Each server can customize its domains. For each group of managing machines tied to a DSMIT file server, you must use a password defined on the file server to operate the DSMIT interface. However, the DSMIT administrator's password does not have to be the same as the root password.

This configuration of DSMIT does not increase the scalability of DSMIT to allow for larger numbers of machines to be managed. Because of the dependency on IP address takeover, only one server in a clustered pair can be actively serving as the file and DSMIT server of its domains.

Alternatively, additional DSMIT servers in a larger cluster can be added and will operate as long as the primary or backup file server is available. Any number of managing machines can be defined, depending upon how many clients are to be administered. However, each managing machine requires a complete DSMIT server license.

Tivoli Management Environment 3.0

Tivoli Management Environment® (TME®) lets you manage systems in a distributed environment. TME uses the hostname/IP address to uniquely identify a machine. Given this implementation, TME 3.0 cannot take full advantage of HACMP's mutual takeover mode to isolate machine failures. TME 3.0 only works with a dedicated hot-standby machine (that is, a machine that takes over the identity of the failed machine).

TME 3.0 addresses the system management problems created by the rapid growth of networks and distributed processing. Its goal is to provide one-touch system management. That is, if you want something to happen on all the machines you manage, you have to push only one button.

Figure 4. Tivoli Desktop Manager

TME 3.0 also reduces the differences in managing disparate hardware and software. The product includes a fairly robust security model that circumvents the traditional UNIX® problem of sharing root-level authority. TME 3.0 provides more granularity for system administration, allowing you to provide various types of permissions on a need-to-have basis. TME also contains a bulletin board on which users can post notices. TME 3.0 provides these features through both a command-line interface and a graphical desktop.

TME 3.0 lets you put sets of machines into a group called a Tivoli Management Region (TMR). These TMRs can be interconnected so that you can make the region layout transparent to the system administrators. This function also isolates node failures to within the TMR; that is, if a server node fails within a TMR, you cannot manage the entire TMR but you can still manage other TMRs.

One of the most important functions of TME is the *profile manager*, a mechanism that lets you establish a relationship between a profile and a set of subscribers to that profile. The subscribers can also be profile managers, thus enabling hierarchical definitions that allow you to easily manage large numbers of machines. Information about how profiles and profile managers are used is in the "Tivoli/Sentry," "Tivoli/Admin," and "Tivoli/Courier" sections.

The purpose of this test case was to verify that TME could be used to manage a cluster of machines. We believe that the cost associated with mutual takeover mode is optimal for a small cluster, so that is the mode we tested. TME 3.0 also should be able to take advantage of HACMP's mutual takeover mode to isolate network and SCSI adapter failures as well as DASD failures.

We installed TME on all machines. One machine in each cluster was a TME server and the other was a TME client. We also defined a profile manager for each cluster that contained all the machines in the cluster. Once a server is installed, all the clients can be installed from the TME desktop on the server. You need the root password and network access to the client machines to perform the client installations.

We set up each cluster (of two machines) as its own TMR (that is, one server and one client). The clusters were then interconnected. The Corporate cluster was connected to all Branch clusters. The interconnection to clusters that had their own system administrators was one-way (that is, Corporate could manage each Branch, but Branch administrators could not manage the Corporate machines). The interconnection to clusters that did not have their own system administrators was two-way.

The goal of Tivoli Management Environment 3.0 is to provide one-touch system management.

High Availability

TME 3.0 was installed on an HACMP file system to isolate SCSI adapter and drive failures. Node failure of the server in a TMR made the region unmanageable, but we could still manage the other TMRs in the unaffected clusters (that is, the Branch clusters).

We encountered a problem implementing the two-way interconnection using the unsecured interconnect method, which forced us to reinstall an entire TMR. To avoid this problem, we recommend the following:

- Always back up your TME database before making significant changes.
- Always back up your TME database after successfully making significant changes.
- Use the secure method to interconnect TMRs.

Manageability

We were able to quickly install and configure TME on every machine in each cluster. The testing went well and enabled us to manage the cluster as planned.

Scalability

This product lets you add a new machine to a cluster fairly easily. Once the machine was added as a client and all the appropriate software installed, the machine was then added to the cluster profile manager. All the profile managers were redistributed, which caused all the appropriate actions on the new machines. However, it was not easy to add completely new clusters (which necessitates adding a new TMR). TMR configurations cannot be copied, which increases the amount of time needed for installations. To add a new cluster, we had to repeat all of the installation and configuration tasks.

We were able to successfully add machines and entire clusters as planned. One procedure that worked well was creating a profile manager that contained all the machines in the cluster. That way, when you add a new machine, you need only add the machine to the cluster profile and redistribute the data targeted for the cluster.

Tivoli/Sentry 3.0

Tivoli/Sentry® (Sentry) sets up event adapters on machines being managed with TME. It is built on top of the TME framework. The adapter can be configured to detect special events (such as a file system becoming full) and take a specific action. Examples of the types of actions that you can configure include running a predefined program, sending mail to a specific person, or posting an event to either Tivoli's Enterprise Console® or TME's Sentry Monitor.

The Enterprise Console collects and manages events on machines in a network. The Sentry Monitor lets system administrators manage events from one location. It is built into the TME desktop and lets you quickly detect events that need attention. The collection mechanism has fairly sophisticated filtering abilities so that specific events can be routed to specific system administrators. It also keeps the events in a shared database so that multiple system administrators can coordinate among themselves to address the events.

Sentry defines profiles that contain definitions of the events to watch, of how frequently to watch, and of what action should take place when an event occurs. You can define these profiles centrally and distribute them to a large number of machines. When these profiles are distributed, the Sentry adapters on each machine read the profiles and react accordingly.

The purpose of this test case was to monitor the events in a cluster. We programmed a profile in Sentry to watch for file systems nearing capacity. Each cluster had a profile manager for monitoring DASD. The profile manager contained a profile with entries for each file system we were monitoring (for example, `/`, `/usr`, `/tmp`). The subscriber to the profile manager monitoring the DASD was the cluster profile manager, which means that the profile was distributed to all machines in the cluster. For hosts with unique file systems, we set up profile managers with subscribers that were specific machines.

We configured Sentry to provide an escalating set of actions depending on how full the file systems were getting. The lowest level of action was to post a notice on the TME bulletin board. The next level of action was to send mail and post urgent events to both the Sentry and Enterprise Console monitors. The highest level of action was to display a warning dialog on a system administrator's desktop.

> ***Tivoli/Sentry sets up event adapters on machines being managed with TME. It is built on top of the TME framework.***

High Availability

Sentry is an add-on application to TME 3.0 and uses TME's high-availability functions.

Manageability

Although we had to enter a quantity of duplicate information when defining profiles, we were able to quickly define the profile and distribute it to every machine in each cluster. Most of our actions worked

well, with one exception: we were unable to forward Sentry events to the Enterprise Console.

Scalability

Sentry is an add-on application to TME 3.0 and uses TME's scalability functions.

Tivoli/Admin 3.0

Tivoli/Admin® (Admin) is a Tivoli product that uses profiles to manage user and group information. Admin is built on top of the TME framework. When the profile is distributed to the target machines, TME data is exported to the appropriate file systems. Backups of the original system files are also maintained so that changes can be backed out.

The purpose of this test case was to manage user and group information. We used Admin to provide user logins on the cluster machines. The profile managers were set up hierarchically so that users could be defined for the entire enterprise or restricted to specific clusters. We used NFS-mounted home directories. We also took advantage of the Admin support to establish E-mail aliases for users in the cluster.

High Availability

Admin is an add-on application to TME 3.0 and uses TME's high-availability functions.

Manageability

We were able to manage user and group information as planned. One problem arose when we accidently distributed the profiles by enabling the option that replaced the profile information (that is, normal user information) on the target machines with information from the database rather than merging it with the profile information on the machines. Because the profiles in the database only included a subset of the normal user information, all the system logins were destroyed. Although not part of our original test case, we were able to verify that you can indeed recover backup versions of the system files.

Scalability

Admin is an add-on application to TME 3.0 and uses TME's scalability functions.

Tivoli/Courier 3.0

Tivoli/Courier® (Courier) uses profiles to identify files to be distributed to machines. Courier is built on top of the TME 3.0 framework. The profiles specify where to get the files and where to put them on the target machines. The profiles can also contain pointers to programs that can be run at various times before or after the distribution of files.

The purpose of this test case was to distribute files to clients in the cluster. We used Courier to distribute the `/.profile` and `/.kshrc` files for root user to all machines in the cluster. In doing so, the root login environment becomes the same on all machines in the cluster.

High Availability

Courier is an add-on application to TME 3.0 and uses TME's high-availability functions.

Manageability

The test case went well and we were able to distribute the files as planned.

Scalability

Courier is an add-on application to TME 3.0 and uses TME's scalability functions.

LoadLeveler 1.2.1

LoadLeveler® lets users run jobs more efficiently by matching their processing needs to available resources. LoadLeveler defines a pool of managing and submit-only nodes. When a job is submitted to LoadLeveler, it is distributed among the nodes (excluding submit-only nodes) based on a user-defined metric.

The purpose of this test case was fourfold. We wanted to determine the ease of adding and deleting nodes, and determine the ease of changing the configuration of the LoadLeveler pool in a clustered environment with other applications configured in the cluster. We also wanted to determine how LoadLeveler's built-in high availability would interact with HACMP. Lastly, we were interested in how LoadLeveler would behave during an HACMP IP Address Takeover (IPAT).

There were no special installation considerations for LoadLeveler.

High Availability

LoadLeveler has built-in high-availability capabilities. It defines alternate central managers in the LoadLeveler pool. LoadLeveler uses a cascading methodology to implement high availability. LoadLeveler also works well with HACMP, except when using IPAT.

To speed up the failover times from the central manager to the alternate central manager during our tests, we needed to add the following lines to the `LoadL_config` file:

```
CENTRAL_MANAGER_HEARTBEAT_INTERVAL = 30
CENTRAL_MANAGER_TIMEOUT = 4
```

Also, for the client to detect the alternate central manager after a failover, we needed to change the following line in the `LoadL_admin` file on the alternate central manager from:

```
SCHEDD_RUNS_HERE = False
```

to:

```
SCHEDD_RUNS_HERE = True
```

We did encounter a significant problem using LoadLeveler with HACMP. We discovered that after a failover in a cluster using HACMP's IPAT, the alternate central manager would not start the `LoadL_negotiator` and `LoadL_collector` daemons. After IPAT, the central manager node was still communicating with the alternate central manager using its HACMP boot address. In this case, the LoadLeveler central manager constantly sends out keep-alive packets to the alternate central manager, instead of the alternate central manager querying the central manager about the status of LoadLeveler. This becomes a problem because the central manager's IP address now resides on the alternate central manager's standby Token-Ring adapter. The clients were now going to the alternate central manager's standby adapter and not finding the daemons running.

To avoid this problem, we recommend that you write an HACMP post-event script for `node_down_local_complete` that would stop LoadLeveler on the central manager node after node failure. Then the alternate central manager can detect it and become the new central manager. However, it will appear to clients that the old central manager is still running because of IP address takeover.

> ***LoadLeveler lets users run jobs more efficiently by matching their processing needs to available resources.***

Manageability

We encountered two problems during testing. The initial version of AFS® installed on the cluster nodes caused all nodes in the LoadLeveler pool to crash. Another problem that affected LoadLeveler was inconsistent IP naming conventions—some nodes had fully qualified hostnames while others did not. To avoid these problems, we recommend the following:

- If you use AFS, ensure that AFS 3.1.1 or later is installed on a system that is included in a LoadLeveler pool.
- Fully qualify the hostnames in the `.rhosts` file on each node.

Other than the problems mentioned above, we were able to successfully use LoadLeveler's job-management functions to submit, manage, and control LoadLeveler jobs.

Scalability

Scalability in LoadLeveler is easily accomplished by editing several stanza files on each node in the pool. We successfully and easily added and deleted nodes, and changed, added, and deleted load metrics.

Interactive Session Support 1.2.1

Interactive Session Support (ISS) lets you manage the distribution of remote sessions.

Clients are transparently connected to servers that have the lowest loads. ISS, using an associated Domain Name Service (DNS) server, dynamically resolves pseudo-host-names that refer to pools of machines.

ISS provides an IP address resolution to clients that points to a minimally loaded machine from the pool. To measure machine loads, ISS can use `rsh` to periodically run user-defined metrics on pool machines, or the LoadLeveler product's load metrics can be used, if available. DNS resources are modified as load conditions in the pool change. You can also define multiple and overlapping pools with different load metrics.

Most installation and configuration work is done on the cluster providing ISS services. Machines that are members of server pools only need to allow `rsh` operations. Client machines only need modifications to the name resolver file `/etc/resolv.conf`.

We installed ISS and DNS applications and data on the cluster's shared disks. A new ISS-specific DNS name service and the ISS process itself were configured as highly available applications. This allowed a second cluster node to take over ISS services upon failure of the first node. An independent metric (not the LoadLeveler metric) was used. For some of the tests we used a simple metric set by the tester to force ISS resolutions to point to the desired machine.

In general, we configured each two-node cluster to be a server pool. For our configuration, we created short scripts to start and stop DNS and ISS. We configured HACMP with ISS and DNS as highly available applications. We then edited the configuration files to define the server pools, load metrics, and IP addresses. Lastly, we altered the clients' DNS configuration fileset.

High Availability

The high-availability aspects of ISS in a cluster were tested with simulated failures of the ISS/DNS node itself. Upon node failure, ISS clients continued to obtain load-leveling connections to the pool. Clients' ISS/DNS requests were then serviced by the takeover node (using IPAT), and client applications continued to connect to unloaded nodes from the pools.

Manageability

The manageability aspects of ISS in a cluster were tested by altering the makeup of the server pools. These tests included changing a pool machine's IP address and changing the metric used to measure machine load. Editing configuration files was all that was required. Clients needed to be modified only if the IP address of the ISS name server changed or if the server pool naming scheme changed.

Scalability

We tested the scalability of ISS in a cluster environment by adding new machine pools, adding machines to existing pools, and creating an overlapping pool that included several clusters and pools. These tests were similar to the manageability tests. The new machine pools consisted of two-node clusters, and an eight-node pool consisting of four two-node clusters. The test case went as planned, and we were able to successfully obtain and use load-managed connections to these pools.

ADSTAR Distributed Storage Manager 2.1.5.6

The ADSTAR® Distributed Storage Manager (ADSM) is an IBM client/server product that lets you consolidate and automate backup and restore functions. ADSM's backup clients support a wide variety of UNIX and Microsoft® platforms.

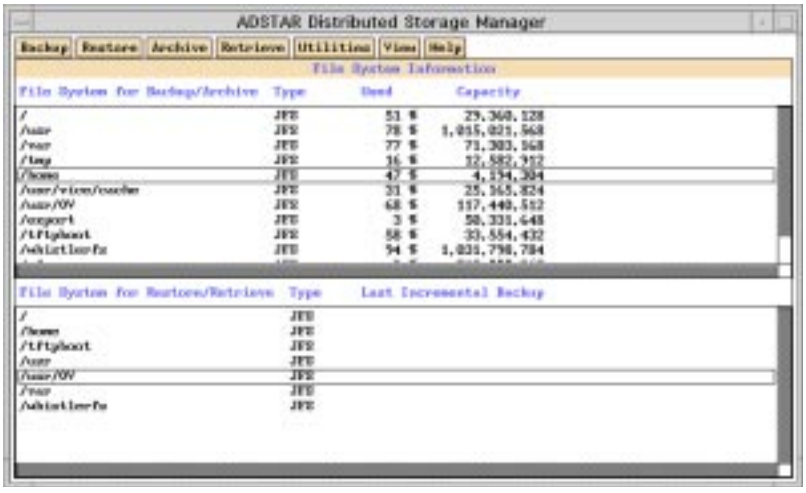


Figure 5. ADSM backup and archive client interface

We followed the standard ADSM installation. ADSM is a complex product with many features for scaling to very large, distributed environments. You should not install ADSM without prior preparation and planning of server and tape storage resources. After becoming familiar with the installation procedures, we were able to configure the ADSM server within a few hours.

High Availability

ADSM has been successfully tested with HACMP by the International Technical Support Center (see the *HACMP/6000 Customization Examples* Redbook, SG24-4498) for one-sided and mutual takeovers. This configuration requires you to use two servers physically connected to a twin-tailed tape storage system. We did not have access to a twin-tailed tape resource, so the purpose of this test focused on providing highly available backup capabilities by installing several ADSM servers in the test environment. We then simultaneously updated the configuration of the ADSM backup clients (using Tivoli's Courier) to allow them to register with a second ADSM server.

We grouped clients together to permanently use 60-70% of the licenses available on their primary ADSM server. The additional licenses could be used by another set of clients if their ADSM servers failed by allowing open client registration. Alternatively, open registration could be allowed for all licenses available on a server. In this case, users would have to keep track of which server they used for backup and restore functions.

Manageability

The ADSM backup client allowed us to easily perform backup and restore functions from any client or server in the network. ADSM can also perform automated backup functions for registered clients. This capability requires you to use a customized Exclude/Include script that specifies the directories and files on the client to be backed up.

We encountered three problems during this test case. The first problem involved basic errors with client environment variables and configuration. The second problem occurred when we registered several clients from the administrator's user interface without fully qualified hostnames, making them unrecognizable to the server when the ADSM backup clients were started on those hosts. The third problem arose when several clients registered remotely with the server and the users manually selected the wrong storage pool.

To avoid these problems, we recommend the following:

- Standardize the environment for the backup clients (we used Tivoli's Courier product).
- Ensure that you configure the hostnames consistently on the ADSM server and its backups.
- If you want to use open registration with the ADSM server, users must know the servers to which to register.

Once we became familiar with the product, ADSM presented no difficulties in operation and significantly eased the backup and restore tasks associated with the other test products. We used ADSM to provide backup functions for nearly all of the products we tested.

Scalability

No scalability problems were encountered during our tests. Several ADSM servers were configured, and we were able to back up the original ADSM server. At the IBM Austin site, ADSM provides on-demand and nighttime backup for several hundred clients.

NetTAPE 1.1

NetTAPE improves and simplifies the management of tape operations and the accessibility of tape devices in RS/6000 network environments. NetTAPE offers consolidated tape operations for all network tape devices from a single user interface, and lets users gain access to remote tape resources transparently, using either a command line or application programming interface.

The purpose of this test focused on allowing global access to tape resources for the servers and clients in our environment. NetTAPE supports a wide variety of devices, from single tape devices to automated tape library servers. For our tests, library servers and twin-tailed tape devices were unavailable for testing.

Initially, several configuration files must be edited to configure the shared tape devices, their hosts, device pools, access control, and any tape libraries. You should carefully read the NetTAPE manual's configuration chapter before undertaking this task. Many of the customization options should be easy to understand for a moderately skilled system administrator.

High Availability

Our tests involved allowing any of the 4x2 cluster nodes and their clients to access six 8mm tape devices spread throughout our environment. The majority of these devices were attached to the HACMP pairs. Because these devices were not twin-tailed, it was not possible to failover the tape drives and recover data from them. However, it was possible to define NetTAPE domains for each cluster node and allow the clients to access an operating domain if the client's primary tape device failed. In real-world operation, a requesting restoration from a failed tape resource would require physically moving the proper tape cartridge to a domain with an operational tape device.

Manageability

NetTAPE allowed easy access to a number of tape devices dispersed in the test environment. We were able to perform tape-management operations, including additions and deletions to the pool of tape devices, with a graphical interface. We could also monitor the status of tape devices.

NetTAPE does require you to define a `.netrc` file for each user. Administrators can use this file to control user access to specific physical tape device hosts. Users must have login access to the tape device host.

Scalability

Our tests showed no problems in allowing users or applications on the server and client platforms to access tape resources over the network. In a local environment, the failure of a single tape device or its host was quickly overcome by simply accessing another available tape drive. In environments where critical backup and restore functions are geographically separated, or where physically relocating a tape cartridge is impractical, a twin-tailed or highly available tape library system would be required.

No scalability limits were discovered using NetTAPE, although our testing did not simulate heavy usage or loading by clients. If your environment has large-scale client tape operations and you want to distribute these functions, you should consider NetTAPE as an option to providing tape devices on every host, or installing and maintaining a tape library server.

Performance Toolbox for AIX 2.1.4

The Performance Toolbox for AIX (PTX) consists of a set of visual monitoring applications (collectively called *the manager*) and data supplier agents that provide performance- and resource-use information from the clients to the manager. The purpose of these tests was

Figure 6. PTX xmperf remote IP monitor

to verify the general operation of PTX in a cluster.

No HACMP-specific procedures were required to install the manager and agents in the test environment.

High Availability

Because the failure of an HACMP node running a manager or agent is likely to terminate the associated processes, the only high-availability capability is the operation of similarly configured managers monitoring the same set of clients. In these tests, the manager could be operating or started on the backup node and acquire information on the set of clients being monitored.

Manageability

The PTX manager allows users to customize the information displayed, which also updates a single configuration file. Once the primary node's configuration was customized, we were able to copy the configuration files onto the backup node.

To aid in managing or monitoring the performance of the clustered nodes, the monitoring application was run on a client in the cluster and various remote resources were monitored on the cluster servers. Various resources could be monitored, including processor, memory, and the network performance of each node. However, be careful when interpreting the clustered nodes' information during a failover. Due to our IPAT configuration (where the backup node assumes the primary's IP address), information from the backup node was reported under the primary node's listing.

Scalability

The PTX Manager can monitor many clients simultaneously. However, the visual representation of the data from a large number of clients can quickly overwhelm a user and become difficult to interpret. System and network administrators using PTX should focus on customizing the PTX displays to report the minimal amount of information needed to monitor a server or client, and troubleshoot specific problem areas or indicators.

AIX 4.1.4 Printing and Print Services Facility 2.1

Print Services Facility™ (PSF™) for AIX is an intelligent printer driver that provides Advanced Function Presentation™ (AFP™) capabilities. PSF supports shared network print server functions and a wide variety of data streams, formats, printers, and platforms.

The purpose of this test case was to verify that basic AIX and PSF printing functions would operate in our cluster environment. When printing in a clustered environment, you must consider several failure issues. When a printer is attached to a clustered RS/6000 platform, both the printer and its host are single points of failure. The same is true for printers attached to terminal servers on a Local Area Network (LAN). For printers with their own network interfaces, the printer and network are single points of failure. In all cases, a clustered print environment would typically require more than one printer resource to be available.

> ***Print Services Facility supports shared network print server functions and a wide variety of data streams, formats, printers, and platforms.***

High Availability

The AIX spooler has been successfully tested with HACMP by the International Technical Support Center (*HACMP/6000 Customization Examples* Redbook, SG24-4498) to create a highly available print system in a cascade configuration. This setup requires you to create remote print queues on each of the cluster nodes and locate printers on non-clustered platforms, terminal servers, or directly on the network. Additionally, the AIX spooling directory is configured on both clustered nodes to reside on a highly available file system.

When clients or servers submit print jobs to a queue residing on the primary cluster node, the job is placed in the highly available spool directory and sent to the remote print server. If a primary cluster node fails, the backup node's queue daemon (`qdaemon`) and remote queue are started. The backup node continues to accept print requests and processes those already in the spool. Print jobs in the process of being submitted when a cluster platform fails are lost and must be resubmitted once the backup has initialized.

To achieve this capability, you must develop two short HACMP event scripts: one that defines the processes the backup must start on takeover, and one that terminates the processes on the backup when the primary node becomes available again.

PSF is not enabled for cascade capability. While some of PSF's capabilities use AIX print-queue concepts, PSF is a much more sophisticated product with functionality well beyond basic printing. If your environment requires highly available print functions, the optimal solution is to have more than one PSF server active (just as most environments have multiple print queues). Clients simply submit print jobs to any available queue, with each print server having multiple queues and printers.

Manageability

Basic AIX print management and PSF resources are easily managed using standard SMIT configuration panels. We were able to successfully reconfigure and update resources in the test environment. Because of the size of some font files used by PSF, you might want to locate these resources on an HANFS platform if you are using multiple print servers.

Scalability

With PSF, you can define multiple print resources and queues. Scaling of resources could entail additional servers, printers, disks, and network resources. For our tests, we reconfigured the Super Branch to be a PSF server, and defined its own printing resources. Printing requests were then serviced by either the Corporate or Super Branch. No problems were encountered in these tests.

Lotus Notes 4.11 and 4.5

Lotus Notes® (Notes) is a Lotus® product for messaging and groupware collaboration. Notes can provide electronic mail, communication, database, document management, and workflow capabilities.

We chose Notes 4.11 and 4.5 because they were the latest releases available when we began our test cases. Notes Release 4.11 does not support the built-in clustering and database partitioning features available in Release 4.5.

In our Notes 4.11 installation, we focused on the operation of multiple servers within our cluster without using HACMP. For the Domino Advanced Services (Notes 4.5), we focused on testing the server mail and replication features in our cluster without using HACMP. For our client operations, we tested the use of Notes executables and resources with HANFS.

High Availability

Existing Notes Server Release 4 solutions have been developed to work with HACMP. For more information, see *Executive Summary: Scalable Lotus Notes on the RS/6000 SP*, available on the Web at the following URL: http://www.rs6000.ibm.com/resource/technology/notesSP/. This document also discusses using the Domino Advanced Services with HACMP.

We installed Notes executables on each server, and placed our data directories and databases on the highly available file systems. The highly available file systems are required when using HACMP, but they are not when using the built-in clustering functions in Domino.

In our installation, the Domino Server automatically enabled the billing functions. You can turn off these functions by removing the billing references in the `notes.ini` files `ServerTasks` field.

Domino comes with its own built-in cluster functions that are dependent upon its database replication. When a server with a replicated database fails, Domino Advanced Services responds to user and application queries by routing those requests to a server with a copy of the needed database. Domino Advanced Services also increases availability

by providing load-balancing features for client requests in a cluster.

HACMP can provide failover capability for mail routing. In environments where mail routing is critical, an HACMP solution is preferable to replicating mail database and routing functions over several servers. Domino Advanced Services does not provide HTTP/IP address failover, which can be provided in an HACMP environment. Alternatively, using LoadLeveler's ISS functions across a Domino cluster with replicated data can create highly available HTTP functions.

If a clustered server replicator is shut down or fails before a replication can be performed, the automatic modification events are lost. For this reason, we recommend that you implement a scheduled replication procedure with all members of the cluster and not depend completely on automatic replication. With HACMP, you can script a forced replication procedure when restarting an HACMP node with the Notes server.

Partitioning allows multiple Domino servers to run on a single platform. While partitioned servers operating on a single host can participate in a cluster, we recommend that other Domino servers in a cluster be located on separate systems. With partitioning, a single server instance can fail while the others continue to operate. Each partitioned server uses its own data directories and resources, requiring additional memory and disk space for each instance of the server.

Installing the Notes client executables and user account on our HANFS cluster allowed clients to operate after a failover. During a failover, the loss of services typically appears to the clients as a standard NFS timeout until the backup platform has completed a takeover. This capability should also be practical in using other file systems, such as AFS or DFS.

If you choose to operate multiple clients on a single server (as we did), you need to set the following variable in the `notes.ini` file for each client and server:

```
Notes_ISOLATION=1
```

Although you can set this variable using the client user's environment files, we recommend using the `notes.ini` file because it is easier to maintain than a user profile or login script.

Overall, we encountered no significant problems operating Notes 4.11 or 4.5 in our cluster. Because Notes uses shared memory functions extensively, we recommend that you implement a script to clean shared memory and release semaphores when a server is restarted. This might prevent or reduce the impact of shared memory problems in your environment.

Domino Advanced Services increases availability by providing load-balancing features for client requests in a cluster.

Manageability

All Notes server-management functions are performed with administration user interfaces. We successfully set up servers, address books, mail routing, and database replication policies without any significant problems. We strongly recommend that you use the Lotus Notes Release 4 on *AIX Systems Installation, Customization and Administration* Redbook (SG24-4694) for planning any installation. Client registrations also presented no significant problems.

For better performance and manageability, we changed the following operating system parameters in our environment:

- Increased the number of processes allowed to 1024
- Expanded paging space to a minimum of 256 MB
- Increased the number of licensed users to 16

We also recommend that each grouping of a Notes' user and mail file system be located in its own volume group and logical

volume. To allow additional resources to be added easily as the number of users grow, groups of Notes users should be broken up into different volumes. Logical volumes can also be mirrored for environments that require it. Ideally, you should place each volume group's log file on a separate physical disk. While this might be impractical for small installations, it reduces the impact of logging activity on other heavily used file systems and eases maintenance tasks.

Finally, centrally locating the Notes client executables and the user databases using HANFS eases maintenance, but does affect performance and can cause client sessions to fail in the event of a prolonged takeover. Alternatively, Notes clients executables can be located on each client (requiring approximately 100 MB of disk space), while user accounts and databases can be maintained on a global file system (NFS, HANFS, AFS, or DFS).

Scalability

A single Notes server's scalability is primarily a function of network and platform performance and the number of clients being supported. Our testing involved creating another Notes server in the test environment and verifying its operation. No problems were encountered in registering or configuring mail routing.

With Notes 4.5, we successfully installed and configured additional Domino servers. To improve both scalability and performance, we recommend that you replicate the most heavily used databases across multiple servers. Partitioning allows additional Domino servers to be located on the same host, and you can add additional resources (CPU, disk, memory) to increase performance.

We did not perform client scalability tests. In any large organization, this would be a function of network, server, and the chosen file system limitations. With middleware applications such as Notes, performance issues become an important concern. We recommend using striped file systems to improve database performance, and that Fast/Wide SCSI or SSA devices be used if possible. Optimizing database locations and replication procedures can also increase performance.

Overall, we recommend that you approach each performance issue separately, without implementing an aggregate of changes simultaneously. Focusing on performance changes one at a time allows you to analyze the effect of each change, and helps isolate and identify the particular bottlenecks in your environment.

Conclusions

Having reached the end of our testing, we have learned that up-front planning plays an integral part in the ease of creating and managing a cluster. We have also learned that some products lend themselves to clustering easily, while others do not. We began with some misconceptions about the number of products that can be used within clusters, and about the costs associated with backup resources. We also arrived at a list of questions that you can ask yourself before you begin clustering (see Figure 7). Our final thoughts cover these topics.

> ***The most important conclusion of our testing is the requirement for detailed planning for any cluster.***

Planning

The most important conclusion of our testing is the requirement for detailed planning for any cluster. While many of the tasks associated with installing and configuring over a dozen products in an HACMP environment seemed daunting at first, hindsight showed that good planning can make these tasks very manageable. Additionally, HACMP itself can seem quite complicated at first, but once we had become proficient with its operation, it was easily maintained, could be quickly configured, and was reliable.

We recommend that administration staff be familiar with all of the products mentioned in this article before attempting to install, configure, and operate them in a

clustered environment. Clustering can complicate the analysis of simple problems that are easily diagnosed in a nonclustered environment. The vast majority of problems we encountered were related to hardware and basic product setup, rather than the result of operating in a cluster.

Products

Several products, not designed for a clustered environment, were successfully configured to work with HACMP. Applications such as DSMIT and NIM can be adapted to operate under HACMP, but if you do not adequately plan for their use in a highly available environment, your customization of the product and use of resources might render them inefficient (or even unusable).

One surprise from our tests was how many of the products could actually be installed and operated from the same cluster. We expected that each cluster would eventually become unmanageable after being loaded with several products, but our experience showed us that clustering does not require a "one-product-per-cluster" model.

For instance, a business using Notes primarily during the day and performing network installs or backup functions mainly at night could reasonably load a single cluster with both capabilities. Other organizations might require only one application to be highly available and not require this of other installed products.

We also had thought that some clustering solutions would be doubly expensive because backup nodes were unused resources until a failover occurs. In our tests, we used many of our backup nodes for operating other products and performing other tasks. For instance, an organization might have a custom, highly available application requiring 24-hour operation, but might have a noncritical need for a second product. The cluster could be configured to run the noncritical application on a backup node, and HACMP could even terminate the other applications to guarantee performance if a failover occurred.

Additionally, disparate hardware and resources can be grouped to create clusters.

Questions to Ask Yourself

We recommend that you ask yourself the following questions before beginning work on your cluster solution:

- How do I prioritize high availability, manageability, and scalability?
- What hardware, software, or tasks require high availability?
- What hardware, software, or tasks do not require high availability?
- What existing hardware or software is compatible with HACMP?
- Do you require your environment to be highly available 24 hours a day?
- To what extent are your system administrators familiar with the products involved?
- What existing equipment is most prone to failure?
- What are the worst-case hardware, software, and resource failures?
- What are the potential single points of failure in your environment?
- What level of performance loss is tolerable in a failover?
- Can nonprimary nodes and resources be used for other tasks?
- What are the growth needs of the environment?
- Does a single operation environment meet all customer's needs?
- How should hardware, software, and resources be distributed to optimize availability, manageability, and scalability?

Figure 7. Cluster-planning questions

Some environments might have rare failover events, and would be willing to tolerate some level of performance loss during these times. These environments can pair less powerful backups with better equipped primary nodes. For instance, instead of creating two clusters out of pairing two J40

Web Pages

The following Web pages can be reached by anyone with an Internet connection:

- HACMP
  http://www.rs6000.ibm.com/software/Appfinder/clustering.html
  http://www.austin.ibm.com/software/Apps/hacmp.html
  http://www.clam.com

- LoadLeveler/ISS
  http://www.ics.raleigh.ibm.com/ics/issfact.htm

- Lotus Notes
  http://www.lotus.com/products/

- NetTAPE
  http://www.rs6000.ibm.com/software/Appfinder/datamanagement.html

- NIM
  http://www.developer.ibm.com/library/ref/about4.1/df4insta.html

- Tivoli
  http://www.tivoli.com/
  http://www.tivoli.com/tivevery/contacts.html

The following Web pages are internal to IBM and can be reached only by IBM employees:

- ADSM
  http://w3.austin.ibm.com/~adsm
  http://w3.austin.ibm.com/afs/austin/depts/itso/itso_web/htmlbooks/sg244601.00/4601fm.html

- DSMIT
  http://w3.austin.ibm.com/afs/austin/depts/itso/itso_web/htmlbooks/gg244380.00/4380fm.html

- HACMP
  http://hacmp.aix.dfw.ibm.com

- LoadLeveler/ISS
  http://w3.austin.ibm.com/afs/austin/depts/k76b/public_html/lxperf/loadleveler/main_load_page.html

- NIM
  http://w3.austin.ibm.com/afs/austin/depts/d57s/nim_html

- PSF
  http://w3.austin.ibm.com/afs/austin/depts/itso/itso_web/htmlbooks/gg243570.01/3570fm.html
  http://w3.austin.ibm.com/afs/austin/depts/itso/itso_web/htmlbooks/gg243570.01/psfdescr.html

- PTX
  http://w3.austin.ibm.com/afs/austin/depts/itso/itso_web/htmlbooks/gg242541.00/2541fm.html
  http://w3.austin.ibm.com/afs/austin/depts/itso/itso_web/htmlbooks/gg242541.00/2541ch6h.html

- Tivoli
  http://w3.austin.ibm.com/afs/austin/depts/tivoli/public_html/index.html

and two 59X models, it might make more sense to have the less powerful nodes serve as the backup resources.

Finding Product Information

For more information about the products described in this article, you can order IBM Redbooks or access product Web pages.

References

The following books can be ordered from your IBM sales representative or, in the U.S., from IBM Customer Publications Support at 1-800-879-2755:

- *ADSM V2 Guide* (SG24-4532)
- *ADSM for AIX: Advanced Topics* (SG24-4601)
- *AIX System Management* (GG24-4380)
- *An HACMP Cookbook* (SG24-4553)
- *Examples Using TME10 on AIX* (SG24-4867)
- *Getting Started with ADSM AIX Clients* (GG24-4242)
- *HACMP/6000 Customization Examples* (SG24-4498)
- *High Availability on the RISC System/6000 Family* (SG24-4551)
- *IBM LoadLeveler Administration Guide—Release 2.1* (SH26-7220)
- *IBM Print Services Facility for AIX: Print Administration* (S544-3817)
- *IBM SystemView for AIX: An Overview* (GG24-2541)
- *Implementing High Availability on a RISC System/6000 SP* (SG24-4742)
- *Lotus Notes Release 4 on AIX Systems Installation, Customization, and Administration* (SG24-4694)
- *Network Installation Management Guide and Reference* (SC23-2627)
- *Printing Under AIX Version 4* (GG24-3570)

Steve Nasypany, *IBM Corporation, 11400 Burnet Road, Internal ZIP 9564, Austin, TX, 78758. E-mail: nasypany@austin.ibm.com. Mr. Nasypany is a staff software engineer in AIX System Management/User Interface. He has a BS in Computer Science from the University of Houston—Downtown.*

Mike Panico, *IBM Corporation, 11400 Burnet Road, Internal ZIP 9564, Austin, TX, 78758. E-mail: panico@austin.ibm.com. Mr. Panico is a software engineer in AIX System Management/User Interface. He has a BS in Computer Science from Florida Atlantic University.*

Sonia V. Weaver, *IBM Corporation, 11400 Burnet Road, Internal ZIP 9561, Austin, TX, 78758. E-mail: soniaw@austin.ibm.com. Mrs. Weaver is a software engineer in AIX Information Design and Development. She has an AAS in and is completing her BLS in Technical Communication at St. Edward's University.*

Juan Zalles, *IBM Corporation, 11400 Burnet Road, Internal ZIP 9564, Austin, TX, 78758. E-mail: juanz@austin.ibm.com. Mr. Zalles is an advisory software engineer in AIX System Management/User Interface. He has a BA in Mathematical Sciences from Rice University.*

A Cup of Java Code

By Peg MacPhail

This article describes a sample Java application exploring Java features such as threads, Abstract Windowing Toolkit (AWT), and the new Event model. The entire code can be found in the Bank Loan Sample Program in this issue.

Click here to go to Sample Program Listings

This sample bank-loan application uses some Java's features, especially threads, that might be useful for a real application. The sample code in its entirety is available in this issue.

Application Background

This program began as a threads programming exercise. Many thread samples are complex number-crunching programs that are extremely difficult to understand. This sample represents a simplistic, rather than realistic, bank-loan application.

Since this program is a Java application, not an applet, it runs independently, outside a browser or an applet viewer. Although Java applications still run on the Java Virtual Machine, they do not have the same security restrictions as an applet. Applications have a main method—not an init method—in the first class that is instantiated to run the program.

The major classes developed for this application are as follows:

- `InputManager`: Handles the user interface
- `BankJob`: Contains the basic context of the loan application, such as how the accounts are named
- `Worker`: Runs on its own thread and pulls `WorkItems` from its own queue
- `WorkItem`: Handles all the actual work it represents since the sample is object-oriented
- `Approver`: Runs its simple algorithm to approve or refuse the loan after all of the transaction work is done
- `Account`: Represents the type of account, such as savings or loan
- `Transaction`: Represents actions associated with each account

How Threads Are Used

Since the `InputManager` runs in its own thread, the main application work does not block the user from doing other work. Although this sample does not take full advantage of this independent thread, your "real" code could. For example, a real program could input loan application request

```
class Worker implements Runnable {
  .
  .
  .
  public void run() {
    while (!done) {
      WorkItem work = getBottomWork();       // get WorkItem
      If ( ( work == null) && (!done) ) {    // if none, steal work
        work = stealWork();
      }
      if ( work != null) {
        work.setWorkMaster(this);
        work.run();                          // make work do itself
        job.ender.decreaseRC();              // decrease release count
                                             // for "final" WorkItem
      }
      else {
        job.threads (id).yield();
      }
    }
  }
  .
  .
  .

class Approver {
  .
  .
  .
  public boolean approveIt() {
    for (int i=0;i<job.NUMTHREADS;i++) {       // Create workers
      job.workers (i) = new Worker (i, job);
    }
    // Create ender (cleanup) WorkItem with release count = 1 so
    // that it will not post itself to the work queue before we have had a
    // chance to add the real WorkItems to the work queue.  Each
    // real WorkItem will add 1 to the release count for our ender
    // WorkItem.
    job.ender = new FinishWork(job.workers*_0*y, 1, job);
    for (int i=0;i<job.COUNT;i++) {
      new ReadAccount(job.workers (0, i, 0, job);
    }
    job.ender.decreaseRC();           // Set release count back
    for (int i=0;i<job.NUMTHREADS;i++) {    // Create threads
      job.threads (i)  = new Thread (job.workers (i);
      job.threads (i).setName("worker" + String.valueOf(i));
      job.threads (i).start();
    }
    for (int i=0;i<job.NUMTHREADS;i++) {   // Wait for threads to end
      try {
        job.threads (i).join();
      }
      catch (InterruptedException e) {
        job.IM.appendText("Thread interrupt" + e);
      }
    }
  .
  .
  .
  }
  .
  .
  .
}
```

Figure 1. Threads example

objects into a work queue for the Approver object(s) to handle.

This sample uses threads with the Worker objects. We used the Blumofe and Leiserson algorithm[1] to schedule the thread work. Each Worker has its own work list of WorkItems, such as reading an account from persistent storage and processing transactions not reflected in the current balance for the account.

To vary parameters with these threads, we built several tunable parameters into this application. Part of the work scheduling algorithm includes a *work number*—the number of transactions a WorkItem is willing to do at one time. If this number is less than the number of transactions to be processed, then the WorkItem creates two new WorkItems, each containing half of the work.

You also can set the number of dummy transactions and the number of threads for each account. The parameters are specified as static class variables in the BankJob object.

To vary the amount of time for reading and for processing a transaction, you can assign a burden value when the program is invoked. This burden value becomes the loop limit in a dummy math calculation that slows down the read or process work.

Two design choices were possible for the Worker object. It could be subclassed from the Thread object and run under a thread, or it could implement the Runnable interface. We chose to have the Worker implement the Runnable interface because it is more generic. Since Java does not support multiple inheritance but does support the implementation of multiple interfaces, we chose to implement the Runnable interface. This choice allows us to inherit from another class in the future, if appropriate. See the sample code in Figure 1.

Global Variables

C and C++ support global variables, but Java does not. However, with Java you can accomplish the same effect by using static class variables, which are either final (constant) or not. The Java Developer's Kit (JDK) uses this capability in several core classes, such as the Color class and its various colors like black, blue, and cyan; or the Math class and its E and PI final static variables. These class variables can be used without instantiating a new object, because the class itself is an instantiated object. The following shows an example of a static class variable:

```
public class BankJob implements
Observer {
  protected static final int
WORKNUMBER=50;
  protected static final int COUNT=5;
  protected static final int
NUMTHREADS=5;
  protected static InputManager IM;
  .
  .
  .
}
```

AWT and the New Event Model

Abstract Windowing Toolkit (AWT) is a set or package of classes used for the Graphical User Interface (GUI).

Figure 2 shows the user interface. The sample application handles the user interface

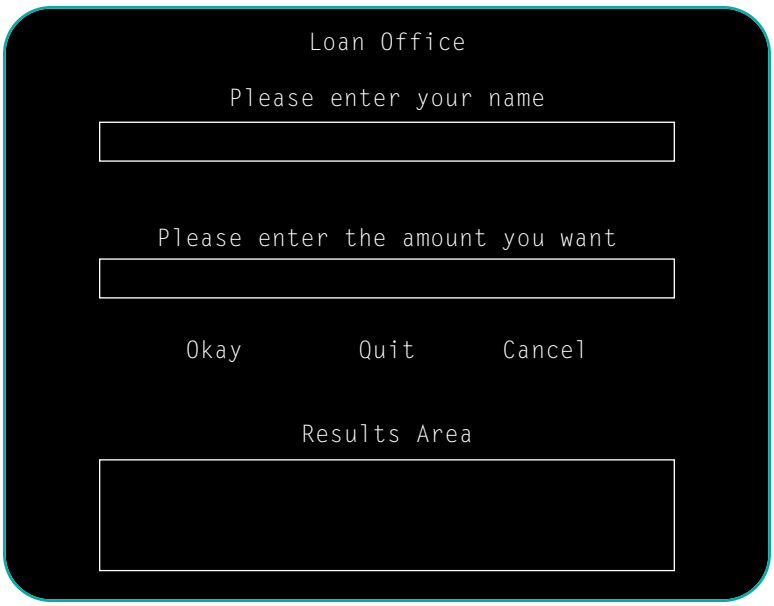


Figure 2. Application user interface

[1]Blumofe, Robert D. and Leiserson, Charles E. "Scheduling Multithreaded Computations by Work Stealing." *Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS)*. Santa Fe, New Mexico. Nov. 20-22, 1994. Pages 356-368.

within the InputManager object. The InputManager uses two different kinds of events: user interface events such as pushing a button, and application events such as obtaining the required input data.

The event model for the JDK 1.1 is a subscription model: the object that is interested in the event registers to receive event notification. The object designer can freely design event handlers. For this sample code, the InputManager registers for the input events of interest, such as pushing the buttons and pressing Enter on the text fields. When the user pushes the "Okay" button or the Enter key, all of the input fields are processed. See Figure 3 for a simple user interface. A more complex interface design could require partitioning the event handling among different objects.

For the application-level events, the InputManager, a subclass of the Observable class, inherits the mechanisms from the Observable class to notify observers. The BankJob object inherits the Observer interface so that it can register for InputManager events, shown in Figure 4. Note that the

```
public class InputManager extends Observable implements Runnable,
                         ActionListener {
  .
  .
  .
  // Constructor
  InputManager () {
  .
  .
  .
    amountField.addActionListener(this);
  .
  .
  .
    loaneeField.addActionListener(this);
  .
  .
  .
    okayButton.addActionListener(this);
  .
  .
  .
    quitButton.addActionListener(this);
  .
  .
  .
    cancelButton.addActionListener(this);
  .
  .
  .
  }
  .
  .
  .
  public void actionPerformed (ActionEvent e) {
    Object source = e.getSource();
    if ( (source == amountField) || (source == loaneeField) ||
         (source ==okayButton) ) {
      processFields();
    }
    else if (source == quitButton) {
           iQuit();
         }
        else if (source == cancelButton) {
               iCancel();
             }
  }
  .
  .
  .
}
```

Figure 3. Action event handling example

```
public class InputManager extends Observable implements Runnable,
                          ActionListener {
  .
  .
  .
  public synchronized void iQuit() {
    setChanged();
    notifyObservers("Quit");
    System.exit(0);
  }
  .
  .
  .
  public synchronized void processFields() {
  .
  .
  .
    setChanged();
    notifyObservers("Ready");
  .
  .
  .
  }
  .
  .
  .
}

public class BankJob implements Observer {
  .
  .
  .
  // Constructor receiving artificial burdens for the worker threads
  public BankJob (int rb, int pb) {
    .
    .
    .
    IM = new InputManager();
    IM.addObserver(this);
    .
    .
    .
  }
  .
  .
  .
  public void update(Observable o, Object arg) {
    if (arg.equals("Ready")) {
      name = IM.getName();
      .
      .
      .
      amount = IM.getAmount();
      .
      .
      .
    }
  }
  .
  .
  .
}
```

Figure 4. Semantic event handling example

`setChanged` method of the `Observable` class must be invoked before the `notifyObservers` method can actually notify observers. Also, note that the `notifyObservers` method can pass an object along with the notification. In this case, we pass a `String` object with the status of the input—either `Quit` or `Ready`.

Conclusion

We hope this sample code will benefit your efforts in creating "real code." Be sure to check the Bank Loan Sample Program associated with this article.

***Peg MacPhail**, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. E-mail: macphail@austin.ibm.com. Ms. MacPhail is a technical consultant for RS/6000 solution providers. She has spent nearly 20 years with IBM in a variety of technical positions. She has a BA from the University of Connecticut at Storrs and a MCS from Texas A&M University in College Station.*

Bank Loan Sample Program

This sample bank loan program consists of several files that belong to the banking package. For this reason, all class files must be located within a directory named banking. Follow these steps to use the program.

1. Create a directory called banking.
2. Copy the `java` and `class` files into the banking directory.

For convenience, you might want to setup an environment variable for your sample class path. For example, on AIX, where the fully qualified path name for your banking directory might be `$JAVAHOME/devcon/samples/classes/banking`, you might set your sample class path name as follows:

```
export MYJSAMPLES=$JAVAHOME/devcon/samples/classes
```

where `JAVAHOME` is set to the directory where the Java Developer's Kit (JDK) is located.

Note: Since the banking loan application is in a package called banking, the Java runtimes expect your class to be prefaced with the package name, for example, banking. See step 4 below.

3. Follow these steps to run the sample program.

   - Unpack the files as appropriate.
   - Set up the appropriate environment variable for your JDK executables, such as `java` and `javac`. On AIX, for example,

     ```
     export PATH=$PATH:$JAVAHOME/bin
     ```

     where `JAVAHOME` is set to the directory in which the JDK is located.
   - Set up the appropriate environment variable for your JDK classes and for this sample (see step 2). Setting `CLASSPATH` for the first time on AIX by following step 2 in setting up an environment variable for your sample classes,

     ```
     export CLASSPATH=$JAVAHOME/classes:/$MYSAMPLES
     ```

     where `JAVAHOME` is set to the directory in which the JDK is located and `$MYSAMPLES` is set to the directory that contains the banking directory.

Note: Do not include banking in the path because that information is presented by the package prefaces included in your code or on the command line.

4. Run the java command as follows: `java banking.Job`.

Note: You must preface the Job class with the package name. The files in this package include:

- `Account.java`
- `BankJob.java`
- `InputManager.java`
- `Job.java`
- `MyMath.java`
- `ProcessAccount.java`

- Transaction.java
- Worker.java
- WorkItem.java

The classes in this package include:

- Account.class
- Approver.class
- BankJob.class
- FinishWork.class
- InputManager.class
- Job.class
- MyMath.class
- ProcessAccount.class
- ReadAccount.class
- Transaction.class
- Worker.class
- WorkItem.class

How the Classes Work Together

The following sections describe how the classes work together.

Job Class

- Instantiated by the `java application` command
- Processes standard input-optional burdens for the worker threads
- Creates the `BankJob` object and calls the `BankJob` run method

BankJob Object

- Creates an `InputManager` object and registers as an Observer with the `InputManager` object created
- Waits for the notification from the `InputManager` that the input is complete
- Creates an `Approver` object and invokes its approve method

Approver Object

- Creates the `Worker` thread objects
- Creates a `ReadAccount` object for each account the person owns
- Creates a `FinishWork` object with a release count equal to the number of `WorkItems` already created. When the other `WorkItems` are done, the `FinishWork` item cleans up
- Waits for the threads to finish
- Executes the loan approval algorithm

Worker Thread Object

- Gets a `WorkItem` off its work queue or steals a `WorkItem` from another worker's queue
- Invokes the `WorkItem`'s run method
- Decrease the release count for the `FinishWork` objects

WorkItems

- Execute the various work-read account transactions and update current balances

Note: If you compile this code, you will get a warning message that the `PrintStream` class has been depreciated.

BankJob.java

The `BankJob` class holds the main logic for processing the bank loan. The `BankJob` object is created by the Job object—this application's main method.

Other sample Java classes in this file include:

- `Approver`
- `ReadAccount`
- `FinishWork` (subclasses of `WorkItem`)

Sample Java classes used from other files include:

- `InputManager`
- `Account`
- `Worker`
- `WorkItem`
- `MyMath`
- `ProcessAccount`

Java classes used include:

- `Date`
- `File`
- `Math`
- `Thread`

Java interfaces used include:

- `Observer`

Features of the sample code include:

- Implementation of the `Observer` interface
- Threads—creation, join
- Error handling—try, catch (`InterruptedException`, `StringIndexOutOfBoundsException`)
- Alternative to global variables—static/final

This application uses the work-stealing algorithm described by Blumofe and Leiserson in "Scheduling Multithreaded Computations by Work Stealing." In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science* (FOCS). Santa Fe, NM. Nov 20-22, 1994. Pages 356-368.

Written by Peg MacPhail, IBM Technical Consultant

to you. IBM shall not be liable for any damages you suffer as a result of using, modifying, or distributing the sample program or its derivatives.

Each copy of any portion of this sample program or any derivative work, must include the above copyright notice and disclaimer of warranty.

BankJob.java

```
package banking;

import banking.InputManager;
import banking.MyMath;
import banking.ProcessAccount;
import banking.Worker;
import banking.WorkItem;
import java.util.*;
import java.io.*;

// This class implements the Observer interface and registers for
// notification by the InputManager which the BankJob object creates.
public class BankJob implements Observer {

  // The following are like global variables for the application
  // Note the use of modifiers "static" and "final"
  protected static final int WORKNUMBER=50;  // Max work chunk for workers
  protected static final int LARGEST=10000;  // Largest transaction amount
  protected static final int COUNT=5;        // Number of accounts
  protected static final int NUMTHREADS = 5; // Number of threads
  protected static InputManager IM;
  protected static Account [] accounts = new Account [COUNT];
  protected static Worker [] workers = new Worker [NUMTHREADS];
  protected static Thread [] threads = new Thread [NUMTHREADS];
  protected static FinishWork ender;         // Ending work item
  protected static String [] acctNames = new String [COUNT];
  protected static String [] acctTypes =
        {"checking","savings", "loan", "IRA", "transactions"};
  //protected static int [] no = {200, 100, 50, 50, 100};
  protected static int [] no = {2, 1, 1, 1, 1};  // Transactions
  protected static int readBurden;           // Artificial load on workers
  protected static int processBurden;        // Artificial load on workers
  protected static boolean workDone;
  protected static boolean inputDone;

  // These variables are for the BankJob object, only
  private static Thread input;
  private static String name;
  private static double amount;

  // Constructor receiving artificial burdens for the worker threads -
  // rb and pb
  public BankJob(int rb, int pb) {
    readBurden = rb;
    processBurden = pb;
    IM = new InputManager();     // InputManager extends Observable class
    IM.addObserver(this);        // BankJob implements Observer
```

```
    input = new Thread(IM);
    workDone = false;
    inputDone = false;
  }

  // Main routine for the BankJob - called by Job object after the BankJob
  // object is created
  public void runJob()  {
    input.start();
    Date begin, end;
    Approver apper;

    while (!workDone) {
      while (!inputDone) {
       waitOnInput();
      }

      IM.appendText("Finished building database.\n");

      begin = new Date();

      apper = new Approver(name, amount, this);

      if (apper.approveIt()) {
      IM.appendText("Loan of " + IM.formatDollars(amount) +
        " approved for " + name + ".  Congratulations!!\n");
      }
      else {
      IM.appendText("Loan of " + IM.formatDollars(amount) +
        " NOT approved for " + name  + ".  Too bad!!\n");
      }

      end = new Date();
      IM.appendText("Done.\n" + printElapsedTime(begin,new Date()));

      inputDone = false;
      IM.reset();
    }
    try {
      input.join();
    }
    catch (InterruptedException e) {
      IM.appendText("Thread interrupt" + e);
    }
  }

  public static String printElapsedTime(Date begin, Date end) {
    long timer = end.getTime() - begin.getTime();
    long front = (long) Math.floor(timer/1000);
    long back = timer - front;
    return ("Time elapsed = " + front + "." + back + " seconds\n");
  }

  public synchronized void waitOnInput() {
    try {
      wait(5000);
    }
    catch (InterruptedException e) {}
  }
```

```
  // This method is invoked by the InputManager when it notifys Observers
  public void update(Observable o, Object arg) {
    if (arg.equals("Ready")) {
      name = IM.getName();
      String fileName = name.replace(' ','M');
      if (fileName.length() > 7) {
      try {
        fileName = fileName.substring(0,7);
      }
      catch (StringIndexOutOfBoundsException e) {}
      }
      amount = IM.getAmount();
      ender = null;
      // Account names are based on the loan applicant's name
      acctNames [0] = fileName+"c";
      acctNames [1] = fileName+"s";
      acctNames [2] = fileName+"l";
      acctNames [3] = fileName+"I";
      acctNames [4] = "transactions";

      buildIt(COUNT, LARGEST);
      inputDone = true;
    }
  }

  // This method with setAccount creates dummy account files
  public void buildIt(int num, int largest) {
    Account a [] = new Account [num];
    for (int i=0;i<num;i++) {
      a [i] = setAccount (acctNames[i], acctTypes[i], no[i], largest);
      a[i].writeOut();
    }
  }

  public Account setAccount (String name, String type, int num, int largest)
{
    Account  a = new Account(name, type, this);
    for (int i=0;i<num;i++) {
      double amt = (double) MyMath.getRandomNumber(largest*100)/100;
      if ( MyMath.getRandomNumber(4) == 2) amt = -amt;
      a.postTransaction(new Transaction (name,amt));
    }
    return a;
  }
}

// This class is a work item for reading the account data
class ReadAccount extends WorkItem {
  ReadAccount(Worker wm, int i, int rc, BankJob j) {
    super(wm, i, rc, "read", j);
    checkToPost();
  }
  public void run () {
   // Read in information for this account
   job.accounts [index] =
          new Account(job.acctNames[index],job.acctTypes[index],job);
   job.accounts [index].readIn();
   // Create a ProcessAccount work item for this account
   ProcessAccount a =
```

```
            new ProcessAccount(workMaster, index, 0, job, 0,
              (job.accounts [index].getTransCount()-1));
  }
}

// This class is a work item for ending the work on the accounts
class FinishWork extends WorkItem {
  FinishWork (Worker wm, int rc, BankJob j) {
    super(wm, 0, rc, "finish", j);
    checkToPost();
  }
  public void run () {
    for (int i=0;i<job.NUMTHREADS;i++) {
      if (job.workers[i] != workMaster) {
      job.workers[i].stop();
      }
    }
    workMaster.stop();
  }
}

// This class implements the approval algorithm for this application
class Approver {
  private String name;
  private double amount;
  private static double loan_factor = 0.15;
  private BankJob job;
  Approver(String n, double amt, BankJob j) {
   name = n;
   amount = amt;
   job = j;
   job.IM.appendText("Getting approval for loan for " + name +
              " for " + job.IM.formatDollars(amount) + "\n");
  }

  public boolean approveIt() {
    for (int i=0;i<job.NUMTHREADS;i++) {
      job.workers [i] = new Worker (i, job);
    }
    job.ender = new FinishWork(job.workers [0], 1, job);
    for (int i=0;i<job.COUNT;i++) {
      new ReadAccount(job.workers [0], i, 0, job);
    }
    job.ender.decreaseRC();
    for (int i=0;i<job.NUMTHREADS;i++) {
      job.threads [i] = new Thread (job.workers [i]);
      job.threads [i].setName("worker" + String.valueOf(i));
      job.threads [i].start();
    }
    for (int i=0;i<job.NUMTHREADS;i++) {
      try {
      job.threads [i].join();
      }
      catch (InterruptedException e) {
      job.IM.appendText("Thread interupt" + e);
      }
    }
    job.IM.appendText("Current balances for " + name + " are:\n");
    for (int i=0;i<job.COUNT-1;i++) {
```

```
      String a = job.IM.formatDollars(job.accounts[i].getAccountBalance());
      job.IM.appendText("   " + job.accounts[i].getAccountType() +
            " balance is " + a + "\n");
    }
    boolean approve = false;
    if (amount < (loan_factor *(job.accounts[0].getAccountBalance() +
                        job.accounts[1].getAccountBalance() +
                        job.accounts[2].getAccountBalance()  +
                        job.accounts[3].getAccountBalance()))) {
      approve = true;
    }
    cleanupFiles();
    return approve;
  }

  // Remove artificial files created
  private void cleanupFiles() {
    for (int i=0;i<job.COUNT;i++) {
      File a = new File(job.accounts[i].getAccountName());
      a.delete();
    }
  }
}
```

InputManager.java

This class manages all input for the banking application.

Java classes used include:

- `Button`
- `Dialog`
- `Frame`
- `Label`
- `Panel`
- `TextArea`
- `TextField`
- `FlowLayout`
- `Font`
- `GridBagConstraints`
- `GridBagLayout`

Java interfaces used include:

- `Runnable`
- `ActionListener`

Features of the sample code include:

- Use of AWT classes listed above
- Event handling—`ActionListener` interface
- Threads—`Runnable` interface
- Error handling—try, catch (`StringIndexOutOfBoundsException`, `NumberFormatException`)

Written by Peg MacPhail, IBM Technical Consultant

InputManager.java

```
package banking;

import java.awt.*;
import java.awt.event.*;
import java.util.*;

// InputManager extends Observable class so that the main processing routine
// can receive notice when the input for a loan application is complete.
// InputManager implements the Runnable interface so that the InputManager
// can run in a separate thread.
// InputManager implements the ActionListener interface so that the
// InputManager can register for and receive input action events.

public class InputManager extends Observable
                    implements Runnable, ActionListener {

  private TextArea resultsArea;     // Output area in display window
  private String name;              // Name of the person applying for loan
  private double amount;            // Amount of loan requested
  private Dialog mainDialog;
  private Panel buttonPanel;
  private TextField loaneeField, amountField;
  private Button okayButton, quitButton, cancelButton;
  private boolean done;
  private boolean disableInput;
  private final int resultsWidth = 40;

  // Constructor
  InputManager () {

     // Initialization of variables
     super();
     name = "";
     amount = 0;
     done = false;
     disableInput = false;

     // Create layout object
     GridBagConstraints gConsts = new GridBagConstraints();
     gConsts.fill = gConsts.NONE;
     gConsts.gridx = gConsts.CENTER;
     GridBagLayout layout = new GridBagLayout();

     Font theFont = new Font ("TimesRoman",Font.PLAIN,14);

     // Create input features
     Frame topFrame = new Frame();
     topFrame.setFont(theFont);
     topFrame.setForeground(Color.blue);

     mainDialog = new Dialog (topFrame);
     mainDialog.setLayout(layout);
     mainDialog.setResizable(true);
     mainDialog.setSize(350,550);
     mainDialog.setFont(theFont);
     mainDialog.setTitle("Loan Office");

     Label loaneeLabel = new Label("Please enter your name");
```

```
    mainDialog.add(loaneeLabel);
    layout.setConstraints(loaneeLabel,gConsts);
    loaneeField = new TextField(20);
    loaneeField.addActionListener(this);
    loaneeField.setEditable(true);
    mainDialog.add(loaneeField);
    layout.setConstraints(loaneeField,gConsts);

    Label amountLabel = new Label("Please enter the amount you want");
    mainDialog.add(amountLabel);
    layout.setConstraints(amountLabel,gConsts);
    amountField = new TextField(10);
    amountField.addActionListener(this);
    amountField.setEditable(true);
    mainDialog.add(amountField);
    layout.setConstraints(amountField,gConsts);

    buttonPanel = new Panel();

    FlowLayout buttonLayout = new FlowLayout();
    buttonPanel.setLayout(buttonLayout);

    okayButton = new Button("Okay");
    okayButton.addActionListener(this);
    buttonPanel.add(okayButton);

    quitButton = new Button("Quit");
    quitButton.addActionListener(this);
    buttonPanel.add(quitButton);

    cancelButton = new Button("Cancel");
    cancelButton.addActionListener(this);
    buttonPanel.add(cancelButton);

    mainDialog.add(buttonPanel);
    layout.setConstraints(buttonPanel,gConsts);

    Label resultsLabel = new Label("Results displayed");
    mainDialog.add(resultsLabel);
    gConsts.gridx = gConsts.CENTER;
    layout.setConstraints(resultsLabel,gConsts);
    resultsArea = new TextArea(20,resultsWidth);
    resultsArea.setEditable(true);
    mainDialog.add(resultsArea);
    layout.setConstraints(resultsArea,gConsts);
}

// This method is called when the InputManager thread is run
public void run() {
    mainDialog.show();
    buttonPanel.show();
    while (!done) {}
}

// This method is called to output information to the display window
public void appendText(String text) {
  if (text.length() <= resultsWidth) {
   resultsArea.append(text);
  }
  else {
```

```
      int i = text.lastIndexOf(" ");
      if (i <= resultsWidth) {
      try {
        resultsArea.append(text.substring(0,i) + "\n" +
           text.substring(i+1,text.length()) );
      }
      catch (StringIndexOutOfBoundsException e) {}
      }
      else {
      StringTokenizer stoke = new StringTokenizer(text);
      String outString = "";
      try {
        while (stoke.hasMoreElements()) {
          String holder = stoke.nextToken();
          if ( (holder.length()+outString.length()) < resultsWidth) {
            outString = outString + holder + " ";
          }
          else {
            if (outString.length()>0) {
            resultsArea.append(outString + "\n");
            }
            outString = holder + " ";
          }
        }
        if (outString.length()>0) {
          resultsArea.append(outString + "\n");
        }
      }
      catch (NoSuchElementException e) {}
      }
    }
  }

  // This method returns the name of the person, if any, requesting a loan
  public String getName() {
    return name;
  }

  // This method returns the amount requested, if any, for the loan
  public double getAmount() {
    return amount;
  }

  // This method is called when the okay button is pressed.  Notice that
  // this method is synchronized so that no one else can access the name
  // and the amount fields while they are being updated.
  public synchronized void okay() {
    if (!disableInput) {
      name = "";
      amount = 0;
      processFields();
      if (amount == 0) {
      resultsArea.append("No amount entered.\n");
      }
      if (name == "") {
      resultsArea.append("No name entered.\n");
      }
    }
  }
```

```
// This method is called when the quit button is pressed.  Notice that
// this method is synchronized so that no one else can access the name
// and the amount fields while quit is in process.
public synchronized void iQuit() {
  setChanged();                          // setChanged is an Observable
  notifyObservers("Quit");               // method which MUST be called
                              // in order for notifyObservers
                              // method to notify observers.
  System.exit(0);
}

// This method resets the input display and input variables. Notice that
// this method is synchronized so that no one else can access the name
// and the amount fields while reset is in process.
public synchronized void reset() {
  disableInput = false;
  name = "";
  amount = 0;
  amountField.setText("");
  loaneeField.setText("");
  loaneeField.requestFocus();
}

// This method is called when the cancel button is pressed. Notice that
// this method is synchronized so that no one else can access the name
// and the amount fields while cancel is in process.
public synchronized void iCancel() {
  if (!disableInput) {
    resultsArea.append("Cancel.  Fields cleared.\n");
    reset();
  }
  else {
    resultsArea.append("Not Cancelled, work in process.\n");
  }
}

// This method gets the data from the input fields - amount, name
public synchronized void processFields() {
  if (!disableInput) {
    name = loaneeField.getText();
    try {
    String amt = amountField.getText();
    if (amt.length() !=  0) {
      Double a = new Double(amt);
      amount = a.doubleValue();
    }
    }
    catch (NumberFormatException e) {
    amount = 0;
    resultsArea.append("Bad amount value.  Please re-enter amount.\n");
    amountField.selectAll();
    }
    if ((name.length() != 0) && (amount != 0) ) {
    appendText("\nLoan request for " + name +
               " for " + formatDollars(amount) + "\n");
    disableInput = true;
    setChanged();                    // setChanged is an Observable
    notifyObservers("Ready");        // method which MUST be called
                              // in order for notifyObservers
```

```
                                // method to notify observers.
      }
    }
    else {
      resultsArea.append("Work in process.\n");
    }
  }

  // This method is implemented for the ActionListener interface
  public void actionPerformed (ActionEvent e) {
    Object source = e.getSource();
    if ( (source == amountField) || (source == loaneeField) ||
       (source == okayButton) ) {
      processFields();
    }
    else if (source == quitButton) {
         iQuit();
       }
       else if (source == cancelButton) {
            iCancel();
            }
  }

  public String formatDollars(double d) {
    double f = Math.floor(d);
    Integer a = new Integer((int)f);
    int r = (int) (100*(d-f));
    if (r<0) r=-r;
    if (r<10) r=10*r;
    String b = "";
    if (r==0) {
      b = "00";
    }
    else {
      Integer x = new Integer(r);
      b = x.toString();
    }
    String out= "$" + a.toString() + "." + b;
    return out;
  }
}
```

Account.java

The `Account` class holds the information related to an individual bank account.

Sample Java classes found in other sample files include:

- `BankJob`
- `MyMath`
- `ProcessAccount`
- `Transaction`

Java classes used include:

- `File`
- `PrintStream`
- `Stack`
- `StreamTokenizer`
- `Vector`

Features of the sample code include:

- Reading and writing from a file
- Storing objects in a Vector and in a Stack
- Error handling—try, catch (`IOException`)

Written by Peg MacPhail, IBM Technical Consultant

Account.java

```
package banking;

import banking.BankJob;
import banking.MyMath;
import banking.ProcessAccount;
import banking.Transaction;
import java.util.*;
import java.io.*;
```

```
class Account {
  private String accountName;
  private String accountType;
  private BankJob job;
  private Vector transactions;             // Contains daily transactions
  private Stack processedTransactions;     // Contains transaction processed
  private double balance;                  // current balance
  private boolean amProcessing;

  Account(String  name, String type, BankJob j) {
    accountName = name;
    accountType = type;
    job = j;
    balance = 0;
    transactions = new Vector();
    processedTransactions = new Stack();
  }
  public synchronized void postTransaction(Transaction t) {
     if (amProcessing) {
       updateBalance(t.getAmount());
     }
     else {
       transactions.addElement(t);
     }
  }
  public void processTransactions(ProcessAccount work, int work_no ) {
    amProcessing = true;
    double subtotal = 0;
    int start = work.getStart();
    int stop = work.getStop();
    if ((stop < transactions.size()) && (start >=0)) {
      if ((stop-start) <= work_no) {
      for (int i=start;i<=stop;i++) {
        Transaction t = (Transaction) transactions.elementAt(i);
        subtotal += t.getAmount();
        processVerify(t.getAmount());
        addProcessed(t);
      }
      updateBalance(subtotal);
      }
      else {
      work.divideWork();
      }
    }
    else
       job.IM.appendText("Bad input to process transactions.\n start ="
           + start + " stop = " + stop + "\n");
  }
  private synchronized void updateBalance(double amount) {
    balance += amount;
  }
  private synchronized void addProcessed(Transaction t) {
    processedTransactions.push(t);
  }
  public synchronized void removeTransactions() {
    while(!processedTransactions.empty()) {
      transactions.removeElement(processedTransactions.pop());
    }
    if (transactions.isEmpty()){
      amProcessing = false;
```

```
    }
  }
  public synchronized void checkPointTransactions() {
    removeTransactions();
    amProcessing = false;
  }
  public String getAccountName() { return accountName; }
  public String getAccountType() { return accountType; }
  public double getAccountBalance() { return balance; }
  public int getTransCount() { return transactions.size(); }
  public void writeOut () {
    try {
      PrintStream ps =
      new PrintStream (new FileOutputStream(new File(accountName)));
      ps.println(accountName + " " + accountType + " " + balance + " ");
      while (!transactions.isEmpty()) {
      Transaction t = (Transaction) transactions.elementAt(0);
      t.writeOut(ps);
      transactions.removeElement(t);
      }
    }
    catch (IOException e) {
      job.IM.appendText("Unable to print to " + accountName);
    }
  }
  public int readIn ()  {
   int r=0;
   try {
     StreamTokenizer toke =
      new StreamTokenizer (new FileInputStream(new File(accountName)));
     r = toke.nextToken();
     if ® == toke.TT_WORD) {
       accountName = toke.sval;
       r = toke.nextToken();
       if (r == toke.TT_WORD) {
       accountType = toke.sval;
       r = toke.nextToken();
       if (r == toke.TT_NUMBER) {
         balance = toke.nval;
         Stack ts = new Stack();
         while (r != toke.TT_EOF) {
           Transaction t = new Transaction ();
           r = t.readIn(toke);
           if (r != toke.TT_EOF) {
             postTransaction(t);
           }
         }
       }
       }
    }
   }
   catch (FileNotFoundException e) {
    job.IM.appendText("File " + accountName + " not found\n");
   }
   catch (IOException e) {
    job.IM.appendText("Error reading in " + accountName + "\n");
   }
   readVerify();
   return r;
  }
```

```
    private void readVerify() {
      MyMath.verifyAccount(job.readBurden);
    }
    private void processVerify(double amt) {
      MyMath.verifyAmount(amt, job.readBurden);
    }
  }
```

Job.java

The `Job` class holds the "main" method for the banking application invoked with `java banking.Job` with two optional parameters for setting work burdens for the worker threads.

Java classes used include:

- `StreamTokenizer`

Features of the sample code include:

- Processing standard input to the main method
- Error handling—try, catch (`IOException`, `ArrayIndexOutOfBoundsException`)

Written by Peg MacPhail, IBM Technical Consultant

Job.java

```
package banking;

import banking.BankJob;
import java.io.*;

// This class is invoked from the command line "java banking.Job" with
// optional "burden" parameters.  The burden parameters are used to
// artificially load down the worker threads by looping through a math
// calculation "burden" times.
// The StreamTokenizer class is used to process the standard input.
public class Job {
  private static final int BURDEN = 1000;
  public static void main(String argv[])  {
    int readBurden = BURDEN;
    int processBurden = BURDEN;
    try {
      if (argv.length > 0) {
      StreamTokenizer toke =
           new StreamTokenizer(new StringReader(argv[0]));
      int r = toke.nextToken();
      if (r == toke.TT_NUMBER) {
```

```
          readBurden = (int) toke.nval;
          if (argv.length > 1) {
            toke =
             new StreamTokenizer(new StringReader(argv[1]));
            r = toke.nextToken();
            if (r == toke.TT_NUMBER) {
              processBurden = (int) toke.nval;
            }
          }
        }
        }
      }
      catch (IOException e) {}
      catch (ArrayIndexOutOfBoundsException e)  {}
      BankJob job = new BankJob (readBurden, processBurden);
      job.runJob();
    }
  }
```

MyMath.java

The `MyMath` class holds the methods for the banking math calculations. Some are dummy methods used to artificially load the worker threads.

Features of the sample code include:

- Static methods that make the `MyMath` class a utility; that is, this class does not have to be instantiated to be used

Java classes used include:

- `Math`

Written by Peg MacPhail, IBM Technical Consultant

MyMath.java

```
package banking ;

public class MyMath {

  // This is a dummy method that performs no useful work except to
  // artificially stress the worker threads
  public static int verifyAccount(int burden) {
     int i, r=0;
     double flier;

     for (i=0;i<burden;i++) {
       flier = Math.log((double)(i+2));
       flier = Math.log((double)(i+1));
     }
     return r;
   }

  // This is a dummy method that performs no useful work except to
  // artificially stress the worker threads
  public static int verifyAmount(double amount, int burden) {
    int i, r=0;
    double flier, a;
```

```
      for (i=0;i<burden;i++) {
        a = (amount - Math.floor(amount))/10;
        flier = Math.exp(7*Math.log(a))/a;
       }

      return r;
     }

    // This method will return a pseudo-random int less than or equal
    // to the value passed in the method call
    public static int getRandomNumber(int largest) {
      int result;
      double f, r;
      double multiplier = Math.max((double)largest,10);
      r = Math.random() * multiplier;
      for (int i=0;(i<5)&&(r<multiplier);i++) {
         r = r * multiplier;
      }
      r = Math.floor(r);
      while (r > largest) {
        f = Math.floor(r/(largest+1));
        r = r - (f*(largest+1));
      }
      result = (int) r;
      return result;
    }

}
```

ProcessAccount.java

This class processes the transactions for an account—getting transactions and changing the balance accordingly.

Other sample code Java classes used include:

- `Account`
- `BankJob`
- `WorkItem`

Features of the sample code include:

- Partial implementation of the Blumofe/Leiserson work-stealing algorithm. If the work is too large—larger than the number of transactions specified by the job `WORKNO`—then the work is divided into 2.

Written by Peg MacPhail, IBM Technical Consultant

ProcessAccount.java

```
package banking;

import banking.Account;
import banking.BankJob;
import banking.WorkItem;

class ProcessAccount extends WorkItem {
  private int start;
  private int stop;
  ProcessAccount(Worker wm, int i, int rc, BankJob j, int str, int stp) {
    super(wm, i, rc, "process", j);
    start = str;
    stop = stp;
    checkToPost();
  }
  public int getStart() {return start;}
  public int getStop() {return stop;}
  public void run () {
    job.accounts[index].processTransactions(this,job.WORKNUMBER);
```

```
    }
    public void divideWork() {
      int cusp = start+((stop-start)/2);
      ProcessAccount a =
         new ProcessAccount(workMaster, index, 0, job, start, cusp);
      a = new ProcessAccount(workMaster, index, 0, job, cusp+1, stop);
    }
  }
```

Transaction.java

This class holds the information for an account transaction.

Java classes used include:

- PrintStream
- StreamTokenizer

Features of the sample code include:

- File I/O
- Error handling—try, catch

Written by Peg MacPhail, IBM Technical Consultant

Transaction.java

```
package banking;

import java.util.*;
import java.io.*;

class Transaction {
  private double amount;
  private String accountName;
  Transaction(String name, double amt) {
    amount = amt;
    accountName = name;
  }
  Transaction() {
    amount = 0;
    accountName = "";
  }
  public double getAmount() {return amount;}
  public String getAccountName() {return accountName;}
  public void writeOut(PrintStream ps)  {
   ps.println(accountName + " " + amount);
  }
  public int readIn(StreamTokenizer toke) throws IOException {
     int r = toke.nextToken();
```

```
      if ® == toke.TT_WORD) {
        accountName = toke.sval;
        r = toke.nextToken();
        if (r == toke.TT_NUMBER) {
        amount = toke.nval;
        }
      }
      return r;
    }
  }
```

Worker.java

This class contains the worker thread object.

Java classes used include:

- `Vector`

Java interfaces used include:

- `Runnable`

Other sample code Java classes used include:

- `BankJob`
- `MyMath`
- `WorkItem`

Features of the sample code include:

- Partial implementation of the Blumofe/Leiserson work-stealing algorithm
- Main thread method—run
- Error handling—try, catch (`IllegalMonitorStateException`, `NoSuchElementException`)
- Synchronization at object level

Written by Peg MacPhail, IBM Technical Consultant

Worker.java

```
package banking;

import banking.BankJob;
import banking.MyMath;
import banking.WorkItem;
import java.util.*;

class Worker implements Runnable {
  private BankJob job;
  private Vector workList;
  private boolean done;
```

```
    private int id;
    Worker(int i, BankJob j) {
      workList = new Vector();
      done = false;
      id = i;
      job = j;
     }
    // This method is called when the thread is started
    public void run() {
      while (!done) {
        WorkItem work = getBottomWork();
        if ( (work == null) && (!done) ) {
         work = stealWork();
        }
        if (work != null) {
        work.setWorkMaster(this);
        work.run();
        job.ender.decreaseRC();        // Job ender WorkItem has a
                                  // releaseCount = # of WorkItems
                                  // in all of the work lists
        }
        else {
        job.threads[id].yield();
        }
      }
    }
    private WorkItem stealWork() {
      int i = MyMath.getRandomNumber(job.NUMTHREADS-1);
      WorkItem  work = job.workers[i].getTopWork(id);
      return work;
    }
    public void postWork(WorkItem w) {
      job.ender.addRC();                  // Job ender WorkItem has a
                                  // releaseCount = # of WorkItems
                                  // in all of the work lists
      synchronized (workList) {
        workList.addElement(w);
      }
    }
    public WorkItem getTopWork(int i) {
      WorkItem w = null;
      try {
        synchronized (workList) {
        w = (WorkItem) workList.firstElement();
        workList.removeElement(w);
        }
      }
      catch (NoSuchElementException e) {}
      catch (IllegalMonitorStateException e) {}
      return(w);
    }
    public WorkItem getBottomWork() {
      WorkItem w = null;
      try {
        synchronized (workList) {
        w = (WorkItem) workList.lastElement();
        workList.removeElement(w);
        }
```

```
      }
      catch (NoSuchElementException e) {}
      catch (IllegalMonitorStateException e) {}
      return(w);
    }
    public void stop() {
      done = true;
    }
  }
```

WorkItem.java

This file contains an abstract base class—`WorkItem`. These `WorkItems` can be posted to the `Worker` queue, when the `releaseCount = 0`. Dependencies are added by increasing the `releaseCount` and removed by decreasing the `releaseCount`.

Other sample Java classes used from other files include:

- `Worker`

Features of the sample code include:

- Use of abstract base class
- Part of the implementation of the Blumofe/Leiserson work-stealing algorithm

Written by Peg MacPhail, IBM Technical Consultant

WorkItem.java

```
package banking;

import banking.BankJob;
import banking.Worker;
import java.util.*;

abstract class WorkItem {
  protected int index;                  // index to in job thread array
  protected Worker workMaster;          // worker object who owns this item
  protected BankJob job;                // overall job object
  private String workType;
  protected int releaseCount;           // count to release this item
  private boolean posted;               // posted to work queue when released
  WorkItem(Worker wm, int i, int rc, String wt, BankJob j) {
    workMaster = wm;
    index = i;
    releaseCount = rc;
    workType = wt;
    job = j;
    posted = false;
  }
```

```
    abstract public void run();
    public synchronized void setWorkMaster(Worker wm) {
      workMaster = wm;
    }
    public synchronized void decreaseRC () {
      releaseCount—;
      checkToPost();
    }
    public synchronized void checkToPost () {
      if ((releaseCount == 0) && (!posted)) {
        workMaster.postWork(this);
        posted = true;
      }
    }
    public synchronized void addRC () {
      releaseCount++;
    }
    public String getWorkType() {
      return workType;
    }
    public String getAccountType() {
      return job.acctTypes[index];
    }
  }
```

AIX Security Certification

By Dinesh Vakharia and Salvatore LaPietra

The German IT security certification authority has awarded the European ITSEC E3 security certificate to IBM AIX. This certification supports IBM's outstanding reputation in IT security.

Business has become increasingly dependent on Information Technology (IT). Today, IT systems must meet specific requirements based on business needs, and provide security and protection for the data and IT services.

IT systems need a well-defined level of security with mechanisms to prevent, detect, and recover from possible harm and damage to the information and IT services. Users must feel confident and trust the accuracy and effectiveness of security functions, such as user identification and authentication, access control, and accounting and auditing. Users must also be able to measure and compare the security capabilities of the IT systems and products to ensure that the systems and products meet their expectations.

Users can develop confidence in the IT system in several ways:

- Trust the vendor
- Rely on vendors to test the IT system or components
- Review the technology and test the IT system internally
- Have an independent organization evaluate the IT system according to a well-defined standard

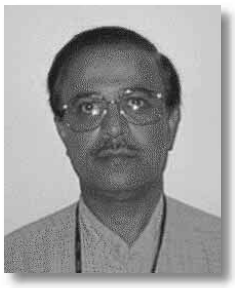
An evaluation based on the IT Security Evaluation Criteria (ITSEC) represents a well-defined level of assurance and confidence in the implementation of the security functions and mechanisms of an IT system.

IBM successfully completed an ITSEC evaluation of the AIX 4.2 Operating System at the E3/F-C2 level of security. An independent evaluation facility accredited by the German government performed the evaluation.

European Security Evaluations Process

Governments have often required products for specific government and military use to have security evaluation and certification. Now, with the increasing dependence on information technologies, businesses also have a heightened need for security and security evaluation.

The European ITSEC established by France, Germany, the Netherlands, and the United Kingdom reflects this concern for security. The European ITSEC defines what is evaluated and the IT Security Evaluation Methodology (ITSEM) provides the description of how the evaluation is performed. Unlike the U.S. Orange Book, ITSEC separates the set of security functions provided by a system from the level of assurance at which these functions are implemented. The final version of ITSEC, published in



Dinesh Vakharia

Salvatore LaPietra

June 1991, proposes ten classes of functions (F-C1, F-C2, F-B1, F-B2, F-B3, F-IN, F-AV, F-DI, F-DC, and F-DX) and six levels of assurance (E1, E2, E3, E4, E5, and E6).

The use of ITSEC has resulted in many security evaluations of products and systems such as operating systems, smartcards, firewalls, antivirus software, and many others.

For AIX certification, the ITSEC schema allowed security functions demanded by corporations to be combined with the E3 level of assurance, which corresponds to the assurance of a B1 system in the Orange Book. E3 enables corporations to benefit from high-assurance systems without the functionality derived from military requirements. Figure 1 shows the evaluation criteria for the U.S. Figure 2 shows the European and U.S.-equivalent criteria.

Trusted Computer System Evaluation Criteria (TCSEC)

Division	Definition	Class	Features and Assurance
D	No Protection	D	No Class in this division
C	Discretionary Protection	C1	Discretionary Protection
		C2	Controlled Access (*)
B	Mandatory Protection	B1	Labeled Security Protection
		B2	Structured Protection
		B3	Security Domain
A	Verified Protection	A1	Verified Design

**AIX certification*

Figure 1. Trusted Computer System Evaluation Criteria (TCSEC) known as the Orange Book

European ITSEC and Related U.S. Criteria

Correctness Level	Definition	Functional Level	
E1	Cumulative Correctness	F-C1	Corresponds to U.S. C1
E2	Configuration Control	F-C2(*)	Corresponds to U.S. C2
E3(*)	Access to Detailed Design and Source Code	F-B1	Corresponds to U.S. B1
E4	Rigorous Vulnerability Analysis	F-B2	Corresponds to U.S. B2
E5	Correspondence between Detailed Design and Source Code	F-B3	Corresponds U.S. B3/A1
		F-IN	Integrity Protection
		F-AV	System Availability
E6	Formal Model and Description and Correspondence between Them	F-DI	Data Integrity during Data Exchange
		F-DC	Data Confidentiality in Communication
		F-DX	Network Security Confidentiality and Integrity

**AIX certification*

Figure 2. European ITSEC and related U.S. criteria

AIX 4.2 Security Evaluation

AIX 4.2 security is evaluated at ITSEC E3 level of assurance and F-C2 function class. The E3 level of assurance is typically found in systems with a security function class of F-B1 or higher. These systems are known as trusted systems, mandatory label systems, multilevel systems, or Compartmental Mode Workstations (CMWs).

The evaluation of AIX on RS/6000 hardware included analysis of the source code and evaluation of the IBM AIX development facility in Austin, Texas, which also complies with the ISO 9000 standard. In addition, site security, the AIX development process, and the distribution procedures were successfully evaluated according to the ITSEC requirements for level E3.

The ITSEC F-C2 function class provides a finely grained, discretionary access control. This makes users individually accountable for their actions through identification procedures, auditing of security-relevant events, and resource isolation.

AIX provides additional security features that map higher classes of function, such as trusted path (F-B2) and access control lists (F-B3). AIX passed extensive security and penetration tests during the evaluation, resulting in an improved level of quality and security. A complete set of security-relevant documentation required for the evaluation is available. Figure 3 shows the AIX certificate.

AIX 4.2 Security Features

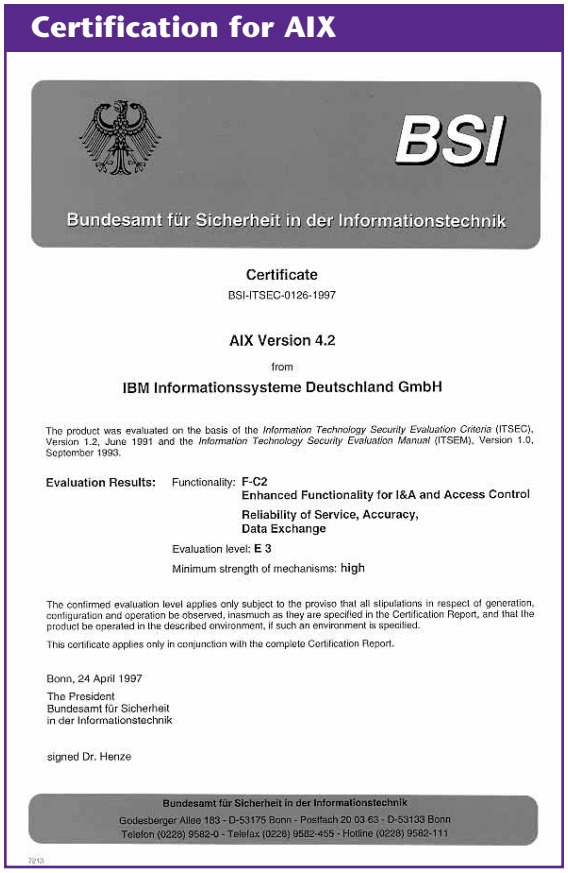
Although hardware protection is available with any RS/6000 system, AIX exceeds the F-C2 security function class with these features:

- Enhanced user identification and authentication
- Data protection through access control lists in addition to the normal UNIX permissions
- Protected password and security databases
- Trusted Path, which prevents applications from trapping the user name and password sequence

AIX passed extensive security and penetration tests during the evaluation, resulting in an improved level of quality and security.

- Login and port control
- User and port restriction capability
- User and port lockout after several failed authentication attempts to log in to the system
- User and port time restriction
- Extensive object-oriented audit subsystem with more than 133 auditable events plus customer configurable events
- Password management
  - —Password minimum age
  - —Password lifetime
  - —Password composition and length
  - —Password reuse
  - —Password triviality checking, including checks against dictionaries of weak passwords
  - —Password expiration warning that can be set
- Security and user management through System Management Interface Tool (SMIT)
- Trusted Computing Base (TCB)
- Integrity of security-sensitive databases, files, and commands
- Trusted shell

Certification for AIX



BSI

Bundesamt für Sicherheit in der Informationstechnik

Certificate

BSI-ITSEC-0126-1997

AIX Version 4.2

from

IBM Informationssysteme Deutschland GmbH

The product was evaluated on the basis of the *Information Technology Security Evaluation Criteria* (ITSEC), Version 1.2, June 1991 and the *Information Technology Security Evaluation Manual* (ITSEM), Version 1.0, September 1993.

Evaluation Results: Functionality: F-C2
Enhanced Functionality for I&A and Access Control
Reliability of Service, Accuracy, Data Exchange
Evaluation level: E 3
Minimum strength of mechanisms: high

The confirmed evaluation level applies only subject to the proviso that all stipulations in respect of generation, configuration and operation be observed, inasmuch as they are specified in the Certification Report, and that the product be operated in the described environment, if such an environment is specified.

This certificate applies only in conjunction with the complete Certification Report.

Bonn, 24 April 1997
The President
Bundesamt für Sicherheit
in der Informationstechnik

signed Dr. Henze

Bundesamt für Sicherheit in der Informationstechnik
Godesberger Allee 183 - D-53175 Bonn - Postfach 20 03 63 - D-53133 Bonn
Telefon (0228) 9582-0 - Telefax (0228) 9582-455 - Hotline (0228) 9582-111

7210

Figure 3. Certification received for AIX

- Customized authentication enabling customers to incorporate and use an alternative authentication mechanism, such as smartcards
- Two-person rule authentication: two users must authenticate to allow a third user to log on
- Single-user level password
- Single sign on for Distributed Computing Environment (DCE)
- User resource limitations
- Security Application Programming Interface (API)
- Journal log of the file system
- Enhanced availability through the mirroring capability of the Logical Volume Manager (LVM)
- Online security documentation through InfoExplorer™

The flexibility of AIX security enables it to grow with your company, because it is compatible with previous and future technology. It is also available on all RS/6000 products. Because the security is integrated, extra packages and recompilation are not necessary. During installation you can choose whether to install TCB to run integrity checks.

Conclusion

AIX now has an ITSEC evaluation and certification that was performed by an independent third party and certified by a governmental authority. Secure and reliable, AIX will enable customers to preserve the availability, integrity, and confidentiality of their information and IT resources. The E3 security certificate makes AIX one of only a few commercial UNIX systems evaluated at this level.

For more information see the IBM AIX Web site at http://www.rs6000.ibm.com or the IBM UBG Security Web site (available to IBM employees only) at http://w3.munich.ibm.com/ubg-security/.

***Dinesh Vakharia**, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Vakharia is a senior programmer/manager in the AIX Security System. His current responsibilities include the functionality of the AIX base operating system security. He holds a Masters in Electrical Engineering from Washington State University.*

***Salvatore LaPietra**, IBM Unternehmensberatung GmbH, Munich, Germany. Mr. LaPietra is an IT security senior consultant with over 10 years experience with IT security. He has a BS in Computer Science from the University of Torino in Italy.*

Broadcast Quality Video Server: MediaStreamer

By Eddie Ho and Sam Juliano

The digital video revolution in the broadcast industry has begun, aided and abetted by the recent orders of the FCC. The FCC has further mandated that by the year 2006, all video will be broadcast digitally, with the last of the remaining NTSC analog broadcasts phased out. Broadcasters must begin adapting their infrastructures to accommodate digital video. The MediaStreamer® solutions provide the building blocks to implement this conversion to the digital domain. With its compatibility with current automation and analog delivery systems, MediaStreamer ensures broadcasters a smooth transition to providing television in the twenty-first century.

In early April 1997, the Federal Communications Commission (FCC) issued a landmark order to the nation's television broadcasters, laying the groundwork for introducing Digital Television (DTV) to the American people. DTV offers consumers the promise of brilliant, high-definition pictures, multiple digital-quality program streams, CD-quality audio programming, and advanced digital services such as data transfer and subscription video.

Under the recent FCC mandate, American broadcasters have been given an aggressive timetable to convert their current analog-based broadcasting systems to digital. Affiliates of the top four networks (ABC, CBS, NBC, and FOX) in the top 10 markets must be on the air with a digital signal by May 1, 1999. Affiliates in the next 20 largest markets must be on the air by November 1, 1999. By that date, 53% of the television households in America will have the opportunity to receive digital video signals.

Digital video represents a significant advancement in quality and capability over the current analog video system. It will include features such as High-Definition Television (HDTV), video-on-demand, and interactive TV. But what will be a huge benefit for consumers initially will be a huge headache for broadcasters, who have millions of dollars invested in analog broadcast equipment and facilities. Broadcasters will have to convert their existing infrastructure to digital technology with as little impact to their ongoing operations as possible.

Broadcasters have many incentives to convert to digital technology. In addition to the potential of new revenue streams from digital technology, cost justifications include increased productivity, higher reliability, fewer miscues, and lower maintenance. For example, many modern broadcast facilities feature arrays of electromechanical Video Tape Recorders (VTRs) being fed by a "sneakernet" system from a large vault of video tape cartridges. This labor-intensive system can be improved

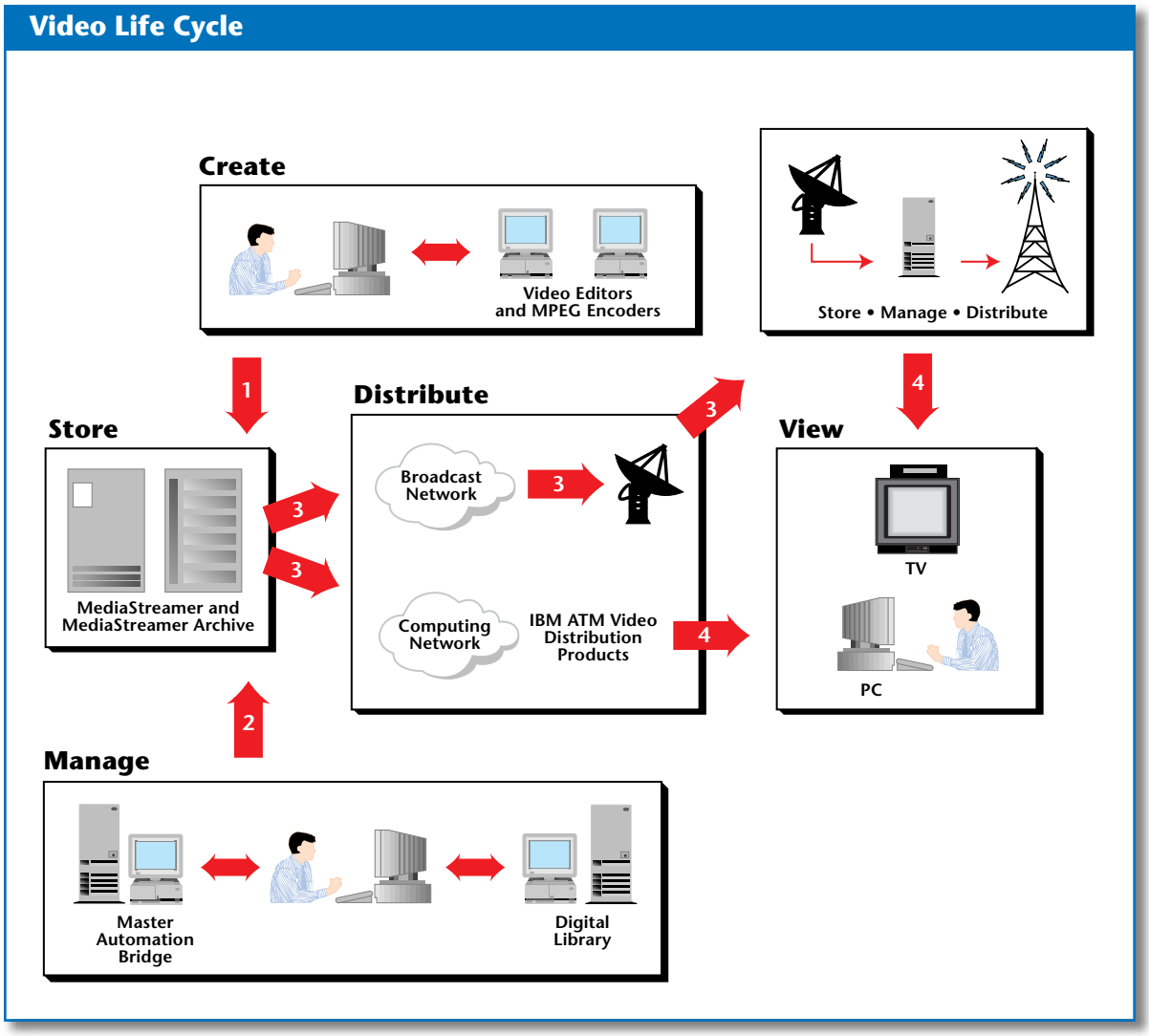


Figure 1. Video life cycle

significantly by the introduction of digital technology, where the VTRs are replaced by a digital video server, and the sneakernet storage and retrieval is replaced by an archive server.

The IBM MediaStreamer solutions are designed to provide broadcasters and other users of broadcast-quality video a system for the efficient storage and distribution of digitally encoded video and audio content. Equally adept at handling both analog and digital outputs, the MediaStreamer can be tailored to meet the needs of traditional broadcast environments while providing the technology for future digital broadcasts. Figure 1 depicts the life cycle of a video product and the MediaStreamer's place in that cycle.

The MediaStreamer solutions are based on IBM RS/6000 workstations enhanced with specialized hardware for video distribution. Expandable, prepackaged systems include two tailored for analog output and two for digital output. In addition, a MediaStreamer Archive solution is available to provide additional storage for facilities requiring larger amounts of data storage.

Industry Solutions

The MediaStreamer solutions are primary elements for video distribution that can meet the needs of various industries. Figure 2 and the following examples demonstrate the versatility of the MediaStreamer in conjunction with appropriate integration technology.

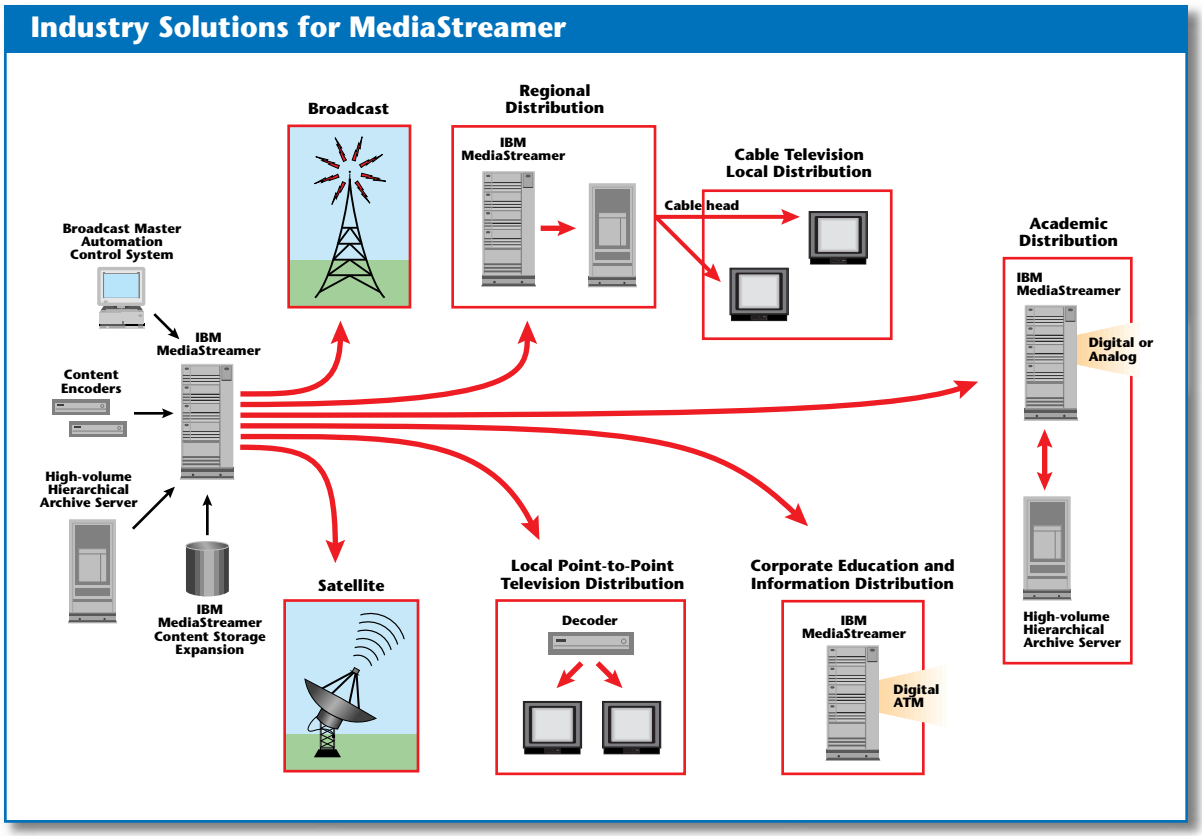


Figure 2. MediaStreamer solutions

High Availability Analog Broadcast. Multiple synchronized MediaStreamers provide a tape machine replacement featuring high availability. Automation bridges attached to the MediaStreamers can detect a failure of a MediaStreamer and initiate automatic switchover to the functional unit. After repair of the failed unit, the bridge initiates resynchronization and updates the contents of the repaired system to match the functional unit. Output from the systems is never impacted.

Distance Learning. A MediaStreamer and MediaStreamer Archive help provide corporate or institutional video data services, such as news on demand and education or training. Video content is created and encoded externally, then recorded on the MediaStreamer. The video data can be saved on the MediaStreamer Archive until needed. A remote client, such as a Video Distribution Module (VDM) attached to the ATM network, can initiate playback of the video data and also view the video data when desired.

Spot Insertion. A MediaStreamer provides cable systems and network station affiliates with the capability to insert video segments into a video feed at specific times. Both short (for example, 10–40 seconds) and long (for example, 2 hours) video files can be handled equally well. Features of the MediaStreamer, such as the genlock output synchronization, ensure that the insertion process is smooth and unremarkable.

Time Zone Delay. The MediaStreamer architecture permits multiple access to a video data file. With this capability, the MediaStreamer provides a broadcaster the ability to schedule separate feeds of the video data at specific times. Even video data that is not completely loaded can be played with the MediaStreamer's "play through" architecture, permitting seven-second censorship delays to be inserted into real-time encoded video feeds.

Scalable Network Operations Center. Network broadcasters can use multiple MediaStreamers and MediaStreamer Archives to distribute different programming to local affiliates. Encoded video data is saved directly to the MediaStreamer Archives as it is prepared. It is then "staged" or moved to the MediaStreamers for playout as needed. Spare MediaStreamers are used for redundancy, and can be hot swapped automatically in case a primary MediaStreamer fails.

MediaStreamer Architecture

The MediaStreamer is designed to stream MPEG-compressed video over ATM OC-3 fiber-optic interfaces and to stream MPEG decompressed video over NTSC and PAL interfaces. Both MPEG-1 and MPEG-2 (see Figure 3) are supported, but ATM output is not limited to MPEG data. Other types of digital data, such as digitized audio or stock prices, can be stored and streamed as well.

Video data files can be loaded in or staged out of a MediaStreamer without disrupting active streams. Both stream and file access are supported. MediaStreamer record and play commands control stream access, whereas read and write commands control file access. Simultaneous stream and file accesses are allowed, even to the same video file. Simultaneous writing and reading is also allowed, providing contentionless playback to multiple outputs while the video data file is being created.

These features result from the application of IBM leading-edge computer technologies to the MediaStreamer architecture. Figure 4 depicts the MediaStreamer and its functional components.

File Storage Manager. This component is responsible for distributing data throughout the disk storage system in such a way to ensure continuous stream operation. Based on the custom Multimedia File System, it is optimized for storage, management, and distribution of large digital video files, and provides guaranteed bandwidth for data delivery. The File/Storage manager responds to asset management commands issued by the Control Server, and coordinates data transfers with the Data Exporter. The File/Storage manager distributes data across disks to optimize bandwidth of the file system. It is responsible for redistributing file system data when disks are added or removed from the MediaStreamer.

Control Server. This component provides the external interface to the MediaStreamer through an Application Programming Interface (API) to clients and

Motion Picture Expert Group (MPEG) Compression Comparison

Compression Scheme	Typical Characteristics
MPEG-1	◆ 352 x 240 resolution ◆ 30 frames per second ◆ l.5 Mbps data rate ◆ Good quality video with some artifacts during fast-moving sequences
MPEG-2	◆ 704 x 480 resolution ◆ 30 frames per second ◆ 6.0 Mbps data rate ◆ Broadcast quality video, particularly at high data rates

Figure 3. Motion Picture Expert Group (MPEG) compression comparison

application programs. The Control Server is responsible for the following:

- Session management: Establishing and closing communications with the MediaStreamer
- Connection management: Determining the status and availability of streaming ports
- Stream operations: Associating ports with video files and controlling streaming, interpreting commands such as start, stop, or pause.
- Asset Management: Managing data files in the MediaStreamer, interpreting commands such as add, delete, or query.

The control server manages the available bandwidth on the MediaStreamer and regulates operations to ensure sufficient resources for all active streams. The control server interprets commands received from the API and directs the File/Storage Manager and the Data Exporter appropriately.

Data Exporter. This component isolates and manages streaming and associated ports. It prepares and provides data to the analog and digital ports. In conjunction with the File/Storage Manager, it supports play and record streams, where data is written to or read from a port. By itself, it supports "pipe streams," where incoming data on a input port is delivered directly to an output port. Using pipe streams, MediaStreamer can convert an inbound ATM stream into an outbound analog format.

MediaStreamer Scalability

The MediaStreamer is designed for scalability: all components of the server can be integrated in a single system for small environments, or can be scaled up using multiple systems with distributed components for a very large environment. Users can grow their configuration by adding more data exporter systems to achieve a higher level of performance with a larger number of data streams. Online disk storage can be

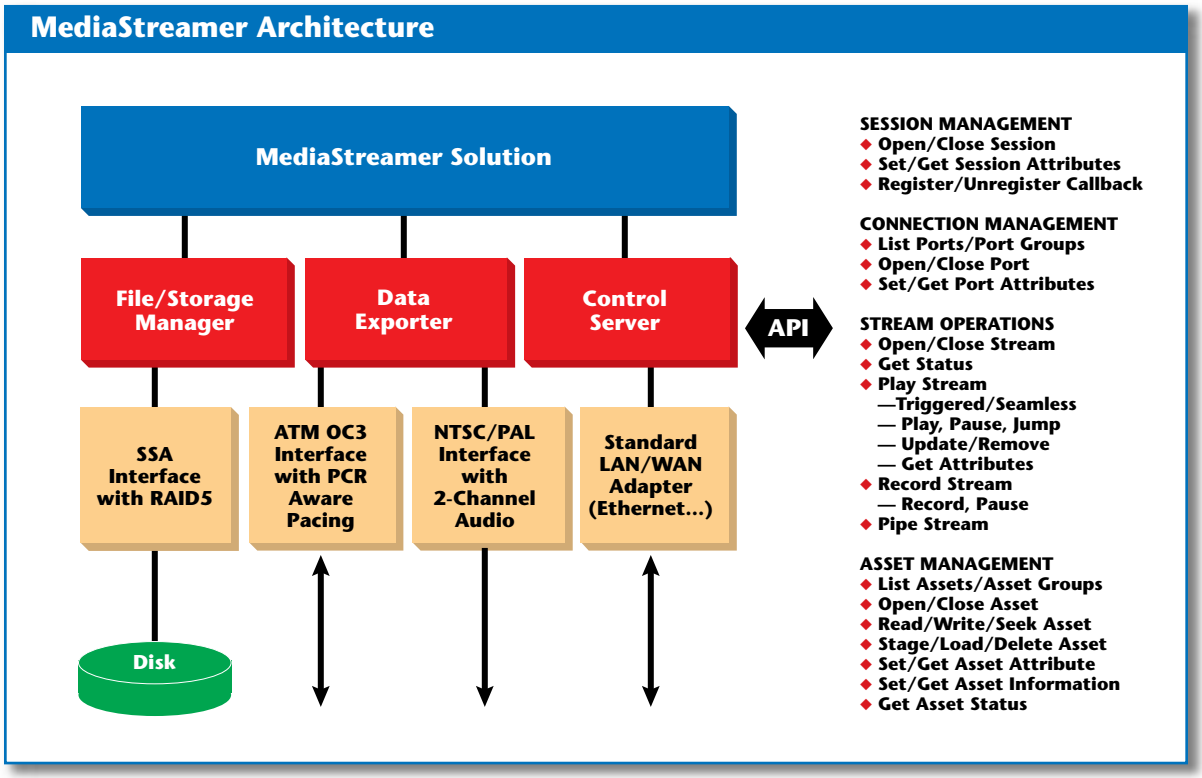


Figure 4. MediaStreamer architecture

MediaStreamer Functional Specifications

All IBM MediaStreamer solutions meet the following functional specifications.

Maximum number of video streams

- 42 separate analog play streams
- 75 separate digital ATM AAL-5 play streams
- 5 separate digital ATM AAL-5 record streams
- 10 separate digital ATM AAL-5 input streams passed through to analog output streams

Maximum video system throughput (Any IBM MediaStreamer can be configured to permit a mix of analog and digital output.)

- Analog play:100 Mbps
- Digital play:120 Mbps ATM AAL-5
- Digital record: 30 Mbps ATM AAL-5
- Analog and digital mixed: 100-120 Mbps (mix dependent)

Video stream rates

- 1.5 to 6 Mbps MPEG-2 transport for analog
- 1.5 to 15 Mbps MPEG-2 AAL-5 for digital

Maximum content loading capacity (Concurrent video content loading is supported. The throughput must be counted in total system bandwidth.)

- ATM TCP/IP: 120 Mbps
- Ethernet TCP/IP: 8 Mbps

Video control

- Deterministic video control command response
- Initial title pre-roll (cue) delay less than 3 seconds
- External trigger option available for frame-accurate starts of analog streams
- Minimum video clip length: 20 second streams for back-to-back unique titles

Figure 5. IBM MediaStreamer

increased to a maximum of 1.242 terabytes with the addition of multiple RAID disk subsystems. Figure 5 shows the functional specifications of a MediaStreamer solution.

Eddie Ho, *IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Ho is a programming consultant in the RS/6000 Executive Briefing Center. He has a BS in Computer Science from the University of Wisconsin and an MS in Computer Science from North Dakota State University.*

Sam Juliano, *IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Juliano is a staff programmer in Internet Multimedia Development. He has a BS in Electrical Engineering from the University of Texas in Austin.*

AIX Questions

The AIX Solution Provider Technical Support Group in Austin, Texas, supports software vendors who are developing or porting applications to AIX. This article is a compilation of questions that are frequently asked by vendors. The name of the responding Technical Support Group staff member appears after each response.

How can I identify any fixes or updates installed on my AIX 4.1 system?

Enter the following commands as root from the root prompt.

```
/usr/sbin/instfix -ic | sort -Au | awk
-F: '{printf \
"%-20s %s\n",$1,$6}' >file

sort -u file >fixesupdates
```

Then `edit` or `vi fixesupdates` to determine what was applied.

—Wade Carlin

Where can I get the latest firmware and System Management Service (SMS) diskette for my PowerPC?

Check the following URL for the latest firmware:

http://www.rs6000.ibm.com/support/micro/download.html

—Wade Carlin

How can I recover from a corrupted hd8 logfile?

To correct a corrupted Journaled File System (JFS) log logical volume, use the logform command to reformat it. The syntax for the command running as root is as follows:

```
/usr/sbin/logform /dev/hd8
```

Answer "yes" when asked if you want to destroy the log.

—Wade Carlin

How do I build shared libraries with C++ code? How can I ensure that static constructors are called? I am building C shared libraries with `ld`; do I need to use `xlC` or some special `ld` option?

To ensure that the static constructors in a shared file are properly initialized, you must construct a library with shared file information in a special way. To do this, use `usr/lpp/xlC/bin/makeC++SharedLib`, which creates a C++ shared library from an export list and a list of object files, and also properly initializes the static constructors. This program outputs a file containing the shared part of the library, which is constructed to handle initialization correctly. Then use the `ar` command to place the file into an archive library.

The C++ User's Guide documents the steps to make a C++ shared library. If there are static constructors or a reference to a shared library with constructors (such as

Wade Carlin

Jeff Simon

iostreams), you must link with xlC to initialize everything correctly. Linking with xlC automatically brings the necessary configuration and library files needed (that is, munch, /usr/lpp/xlC/lib/crt0.o, /usr/lpp/xlC/lib/gcrt0.o). Be sure to look at /etc/xlC.cfg, which will automatically bring in these files.

—Jeff Simon

What rules govern whether source code is compiled for C or C++?

The following rules can differentiate whether the C or C++ compiler is used:

1. All files with the suffix .c (lower case c) are assumed to be C files unless rule 3 applies.

2. All files with the suffix .C (upper case C) or suffix .cpp are assumed to be C++ sources.

3. Option -+ indicates that all files, regardless of the suffix, are assumed to be C++ sources.

—Jeff Simon

What is a zombie process?

A zombie process occupies a slot in the process table, but has no other space allocated to it in either user or kernel space. The process table slot that it occupies is partially overlaid with time-accounting information to be used by the times subroutine. (See the sys/proc.h file.)

A process remains a zombie until its parent issues one of the wait subroutines. At this time, the zombie is laid to rest (deleted) and its process table entry is released. If the child is killed but the parent is not notified of this event, then the child will remain in a zombie state indefinitely until the parent exits, at which time it will be inherited by the init process that has a pid of 1.

—Jeff Simon

How can I turn off keyboard clicking?

Gordon Thagard

There are three ways to approach this problem:

1. From CDE, open the Style Manager, select Keyboard, then turn off the keyboard volume.

2. From System Management Interface Tool (SMIT), start SMIT with smitty chgkdb and turn off the keyboard volume.

3. From the command line, type xset -c.

—Gordon Thagard

How can I install Base AIX on a PowerPC from a CD-ROM?

Follow these steps:

1. Insert the CD-ROM.

2. Reboot the machine.

3. After the boot sequence, press the PF1 key to enter System Management Services.

4. Insert the SMS diskette, if necessary.

5. Go to the Startup menu and make the CD-ROM drive the first boot device.

6. Exit from SMS and continue booting from the CD.

7. Setup the Install environment according to preferences <console/language>.

8. At the Installation & Settings screen, choose option 1 to define install preferences.

9. Choose Preferred Install Type and add any hdisks not in the install list (at the user's discretion).

10. Choose option 0 when done. AIX will now install from CD 1. After AIX installs, the system administrator

must decide which additional Licensed Program Products (LPPs), if any, will be installed.

—Gordon Thagard

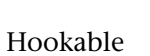
How can I make an SLHS application run on AIX 4.1.5 or 4.2?

An executable Shared Library Hookable Symbol (SLHS) application written on AIX 4.0 through 4.1.4 must be changed slightly to run on AIX 4.1.5 or 4.2 as follows:

- To run SLHS application on AIX 4.1.5, install PTF U447112. U447112 updates `bos.rte.libc` to 4.1.5.5 corresponds to SLHS RTE 1.1.6.7.
- To run SLHS application on AIX 4.2.0, install PTF U446665. U446665 updates `bos.rte.libc` to 4.2.0.8 corresponds to SLHS RTE 1.1.6.7.
- If you want to rebuild (compile and link) a complete SLHS application written in AIX 4.0 through AIX 4.1.4 to run on AIX 4.1.5 and 4.2., then you should be aware of the following:

  Since SLHS is not supported on AIX 4.2, you must use the native runtime linking and dynamic loading library on AIX 4.2. In general, the sv compiler will become the normal flavor (for example, `xlc`), and `svld` becomes `ld`. The `ld` option `-G` will build modules that require runtime linking, and the `-brtl` option will build main applications that use these modules. The library option `-ldl` must also to be added to resolve calls to `dlopen()`.

—Hung Dinh



Hung Dinh

How can I determine if an archive file (.a) is a shared library?

AIX archive files can contain both shared and non-shared objects. An archive is considered shared if it has at least one shared object. Using the `dump` command, you can determine whether an archive file contains shared objects.

Issuing `dump -vT mylib.a` results in the following message if no shared objects are in the archive:

```
Loader section is not available
```

—David Carew

David Carew

Compiled by ***Wade Carlin and Jeff Simon****, IBM Corporation, 11400 Burnet Road, Austin, TX 78758.*