Netra j 2.0.1 Product Notes

THE NETWORK IS THE COMPUTER™

Sun Microsystems Computer Company
A Sun Microsystems, Inc. Business
901 San Antonio Road Palo Alto, CA 94303-4900 USA
650 960-1300
fax 650 969-9131

Part No.: 805-5915-10
Revision A, Month 1997

Please
Recycle

Contents

Netra j 2.0.1 Product Notes

This document contains late-breaking product information for the Netra j 2.0.1 software release.

Additional late-breaking information on the Netra j 2.0.1 software and JavaStation™ computer will be posted on the Web at `http://www.sun.com/javastation`.

What's New in This Release

The Netra j 2.0.1 software contains the following enhancements to Netra j 2.0:

- *NC Compliance* – The JavaStation client software included in Netra j 2.0.1 is compliant with industry standards for network computers.
- *Netra j Administration Enhancements* – Network Computer Server Administration and NIS (Network Information Service) configuration have been improved.
- *New JavaOS™* – The JavaOS 1.1.1 (aeh) binary features improved garbage collection, accelerated graphics, and numerous bug fixes.
- *New HotJava™ Packages* – HotJava Browser 1.1.4 and HotJava™ Views™ 1.1.3 provide bug fixes and improvements, including better memory management and a reduced runtime footprint. Most of the bug fixes involve improvements to Web page rendering in order to comply with industry standards.
- *GO-Joe Software Update* – GO-Joe software for network computer access to X Windows applications has been updated and improved. For details on GO-Joe version 2.01a2, see the file `RELNOTES-GOJOE.TXT` in the `docs` directory on this CD-ROM.
- *OC://WebConnect Pro Software Update* – OC:/ /WebConnect Pro software for 3270 legacy connectivity has been updated and improved. Enhancements include:
  - Token authentication of clients

- Improved user interface for session startup
- Better printing capability

OC:/ /WebConnect 3.2.4.1 includes OpenVista 1.0 for rejuvenating classic 3270/ 5250 green-on-black screens. For details on OC:/ /WebConnect, see the file `RELNOTES-WC.TXT` in the `docs` directory on this CD-ROM.

- *Static Linking Kit (SLK)* – The SLK is a kit that allows administrators and OEMs to bundle their own applications into JavaStation flash memory as the main application.

Netra j 2.0.1 Software

This section provides notes on the Netra j 2.0.1 software.

Netra j Software Disk Space Requirements

The Netra j software packages are not relocatable. Be sure that you have enough space in `/opt`, `/usr`, or `/var` file systems. TABLE 1 lists the approximate disk space requirements.

TABLE 1 Netra j Software Disk Space Requirements

File System	Space in Mbytes
`/`	21
`/opt`	83
`/usr`	5
`/var`	3

System Reboot

Caution – You *must* reboot the server after installing the Netra j 2.0.1 software and before you begin setup.

Fresh Installation

On page 18 of the *Netra j 2.0 Installation Guide*, under Initial Configuration, you are no longer required to complete steps 3 through 9. The Netra j 2.0.1 software will not display these configuration pages.

Netra j Administration Interface

HotJava Views administration requires HotJava Browser for the Solaris operating environment. HotJava Browser 1.1 is included with the Netra j 2.0.1 software. For all other Netra j 2.0.1 administration tasks, any industry-standard browser can be used.

The HotJava Views administration applets use JDK™ 1.1 application program interfaces (APIs), including AWT 1.1 (abstract windowing toolkit) and the New Event Model. Therefore, HotJava Views administration requires a fully JDK 1.1-capable browser. As of this writing, HotJava Browser 1.1 for the Solaris operating environment is the only browser that supports AWT 1.1 and New Event Model features in JDK 1.1. Netscape Navigator™ 4.03 supports only a subset of JDK 1.1. If you prefer a Netscape browser, check the Netscape web site at `http://devedge.netscape.com/software/index.html` for the latest information about AWT 1.1 and New Event Model support.

Before configuring HotJava Views Client Administration, you must set preferences for applet security to Medium.

▼ To Set Applet Security in HotJava Browser

1. **At the top of HotJava Browser, select Edit ➤Preferences ➤Applet Security.**
2. **Set signed applet security to Low, and set unsigned applet security to Medium.**

Netra j Software Package Additions

TABLE 2 lists the Netra j software package additions to accommodate the high encryption and low encryption software.

TABLE 2 Netra j Software Additions

Current Package ID	High-Encryption Package ID	Low-Encryption Package ID
`SUNWjsos`	`SUNWjsosh`	`SUNWjsosl`

Documentation

The following documents for Netra j 2.0 support the Netra j 2.0.1 software.

- *Netra j 2.0 Installation Guide* (805-3080-10)
- *Netra j 2.0 Administrator's Guide* (805-3076-10)

For updates to the Netra j 2.0 documentation, see page 34.

Netra j Software Management Module Error

When the Netra j Software Management module is used to make additions to or removals from the system, the Sun™ WebServer™ may time out. This results in an error message from the browser.

For HotJava Browser:

```
Exception: java.net.SocketException
```

For Netscape Navigator:

```
500 server error
```

If this error occurs, make sure that the software component (package, patch, or cluster) was fully installed or removed. The install or remove process may continue to run in the background after the browser returns with an error.

Make sure you either add or remove software components one at a time or use the `admintool` or `pkgadd` utility to add or remove these components.

Netra j Software Management Tool

When Lotus eSuite is installed on the Netra j server, the following error messages are reported:

```
Connection error
Something went wrong while reading the document
http://localhost:81/cgi-bin/uncgi/misc/sw-admin/cgi-bin/
process_install.cgi?type=package
Either the connection was closed prematurely or some other
exception occurred
Exception: java.net.SocketException
Socket closed
```

However, observing the console window shows that the installation finishes successfully.

Netra j Save and Restore Feature

Some of the administration information that is entered using the Network Computer Administration forms is not automatically saved as part of the Netra j Save and Restore feature. The relevant system files must be saved manually if you intend to restore these settings after a system crash. The entire directory `/var/dhcp` should be saved to an off-line storage medium such as magnetic tape or diskette. It is recommended that the contents of this directory be backed up at regular intervals, particularly when new NC's have been added, deleted or modified using the Netra j user interface. These files must be restored separately in addition to the default restore procedures required during a crash recovery of a Netra j server.

Configuring Users' Home Directories

The home directories of user accounts set up for JavaStation users must be NFS-shared. This requirement is handled automatically when users are added or modified using the Netra j User Accounts module. Some user accounts may not be NFS-shared if they existed prior to Netra j installation/upgrade. To activate NFS-sharing of such user accounts, follow the instructions below.

▼ To Activate NFS-Sharing of Existing User Accounts

1. **Become root.**
2. **Add the following line to the** `/etc/dfs/dfstab` **file for each user:**

```
share -F nfs -o rw -d “Home Directory” /export/home/username
```

where `/export/home/`*username* is the user’s home directory.

If existing users all have a common home directory base, for example, `/export/home`, you can edit the `/etc/dfs/dfstab` file as follows:

```
share -F nfs -o rw -d “Home Directory” /export/home
```

Using Java Webserver with a Netra j Server

If you intend to use Java Web Server 1.1 as your default web server, it may not successfully establish itself as the default web server on port 80 when you boot the Netra j server. Instead, the Sun WebServer (bundled with the Netra j 2.0.1 software) substitutes itself as the primary web server running on port 80. As a workaround, you must add `sleep 5` to the script `/etc/rc3.d/S95http`.

```
case “$1” in
‘start’)
    sleep 5
    if netstat -na | grep LISTEN | grep -w 80 >/dev/null
    then
        echo “$SCRIPT_NAME: httpd not started, port in use”
        exit
...
....
....
```

This adjustment provides an adequate interval for the Java WebServer to initialize itself during bootup and prevents Sun WebServer from monopolizing port 80.

DHCP

Sending a `SIGHUP` to the DHCP server daemon causes the `in.dhcpd` file to reread its data during the idle interval, anywhere from 0-60 seconds. The moment the `in.dhcpd` file actually rereads its data during the idle interval depends on where the DHCP server is in its polling cycle. For busy servers, you should run `/etc/init.d/dhcp stop` then `/etc/init.d/dhcp start` to force the data to be reread.

Web Proxy

On networks with both a Netra j server and a separate Web proxy server, you must modify JavaStation HotJava Views and HotJava Browser preferences to allow *no proxy for* the Netra j server. Otherwise, the JavaStation client always contacts the Web proxy for all its services, which could lead to potential connectivity problems.

Upgrading to Netra j 2.0.1 Software

This section describes Netra j 2.0.1 software upgrade considerations.

Upgrading to HotJava Views 1.1.3

HotJava Views 1.1.3 is automatically installed with the Netra j 2.0.1 software. During the HotJava Views upgrade, the following information may be lost:

- Groups you have configured
- Application icons you have added to Selector

To preserve group configurations, copy the file `/opt/SUNWjdt/lib/props/jdt.group` to a different directory prior to the upgrade, or save it under a new name such as `jdt.group.save` (filenames unknown to HotJava Views will be ignored during the upgrade). After the upgrade, you can replace the new `jdt.group` file with your saved file, as the file format has not changed with 1.1.3.

Due to format changes in the `/opt/SUNWjdt/lib/props/selector.apps` file, it will be necessary to add application icons back to Selector after the upgrade. For instructions, see “Adding an Application Icon to the HotJava Views Selector” on page 21.

As with any upgrade, it is a good idea to back up the file system in case you wish to retrieve old files after the upgrade.

Upgrading From Netra i to Netra j

Upgrading from a Netra i system to a Netra j system means a shift of emphasis in the capabilities of the server.

- Netra i systems provide only NIS client capabilities, whereas Netra j servers provide NIS server capabilities. For example, a Netra i site with local user accounts must become NIS user accounts for login through JavaStation systems.
- ISDN Internet access, considered central to a Netra i server, is no longer available to the Netra j server.

Netscape Navigator Requirement

Netscape Navigator is required for systems upgrading from Netra i versions 3.1 and 3.2 that support Netscape Enterprise Server™ and FireWall First!.

Local Name Service

The following information affects systems upgrading from Netra i 3.1 and Netra j 1.1.

Before completing the initial configuration steps on the Netra j 2.0.1 software, check the Netra j host name from the Local Name Service module, under *Name Services*.

If the Netra j host name is mapped to the loopback IP address (`127.0.0.1`), you must delete this reference to the host name machine. In such cases, you must add a separate host-to-IP address mapping that maps the host name of the Netra j server to the IP address of the network interface that is used to provide boot services to the JavaStation client or network computer.

File System Backup

When you upgrade from Netra i 3.1 to Netra j 2.0.1, the file system backup options set for Netra i 3.1 are lost after the upgrade. The system administrator should restore the options for Netra j 2.0.1.

Upgrade With OS Installation

If you are upgrading Netra i 3.1 and Netra i 3.2 systems to Netra j 2.0.1 and you must load the operating system, make a tape backup of the entire file system. In particular, back up the Netscape Web Server port 80 `htdocs` directory and subsequently restore it once the upgrade process is complete and the `SUNWnse` package reinstalled.

For Netscape Enterprise Server 2.0, this directory is `/usr/local/netscape/nse-home/docs;` for Netscape Enterprise Server 3.0 it is `/opt/netscape/suitespot/docs`.

Upgrading From Netra j 1.1 Software

Administration Web Server User Password

The port 81 Administration Web Server user password is lost on upgrading to Netra j 2.0.1 from Netra j 1.1 because the National Computer Security Association (NCSA) server is no longer used for Netra j administration. Sun WebServer now performs this task, and the Administration Web Server user password is stored differently. This is why you are always prompted for a new Administration Web Server password when logging in immediately after an upgrade.

Access control lists associated with the NCSA server are also lost. You must apply similar access rights to the server through Sun WebServer after upgrading to Netra j 2.0.

Local Name Service

The following information affects systems upgrading from Netra i 3.1 and Netra j 1.1.

Before completing the initial configuration steps on the Netra j 2.0.1 software, check the Netra j host name from the Local Name Service module, under *Name Services*.

If the Netra j host name is mapped to the loopback IP address (`127.0.0.1`), you must delete this reference to the host name machine. In such cases, you must add a separate host-to-IP address mapping that maps the host name of the Netra j server to the IP address of the network interface that is used to provide boot services to the JavaStation client or network computer.

JavaStation Client Software

This section contains notes pertaining to the software that will run on the JavaStation client machine.

Localization Directory Change

The directory `/export/root/javaos/locale` has been changed to `/export/root/javaos/classes`.

Updating the `javaos` binary

After updating a new `javaos` binary to flash memory, the server automatically reboots the JavaStation and the JavaStation may display the following warning:

```
Can't open device.
Keyboard not present. Using ttya for input and output.
```

This warning is displayed when the JavaOS software is first installed or when the end-user elects to update the operating system on reboot.

This warning is from the Open Boot Prom (OBP) and has no impact on how JavaOS functions.

Developer Documentation

Information on APIs (Application Programming Interfaces) for developers building applications to run on JavaOS will be posted at `http://www.sun.com/javastation`.

Modem Dialup from the JavaStation Computer

The JavaOS™ operating system supports Point-to-Point Protocol (PPP) modem dialup from the JavaStation computer to a PPP server.

The JavaOS PPP Dialer window appears automatically on the JavaStation monitor if the user turns on the JavaStation when its Ethernet cable is disconnected. The user can click Connect to dial immediately or Options to set or modify dialup options.

FIGURE 1 JavaOS PPP Dialer Window

Complete modem dialup instructions for the JavaStation are in the JavaStation User Setup Forms, which are available:

- As an appendix in the *Netra j 2.0 Administrator's Guide* (in SGML format)
- On the documentation page of the Netra j 2.0.1 administrative interface (in HTML format)

Statically Linking an Application to JavaOS

One method of delivering a user application to JavaStation clients is to bind the user application files to the JavaOS binary delivered to JavaStation clients at bootup. When JavaOS boots, the user application is started automatically. This procedure is enabled by the Static Linking Kit (SLK), a special JavaOS binary file to which you can link your own application files.

To use the SLK, you must prepare a directory tree containing JavaOS files and application files and then run the `link_javaos` utility to create a new build of JavaOS incorporating the application files. This procedure is described below.

Requirements

Running the SLK requires the following Solaris packages:

- `SUNWarc`
- `SUNWbtool`

- SUNWcsu
- SUNWlibm
- SUNWsprot
- SUNWtoo

▼ To Statically Link a Custom Application to JavaOS

1. **Run** `pkginfo` **to make sure the required packages are installed on your Solaris system:**

```
% pkginfo SUNWarc SUNWbtool SUNWcsu SUNWlibm SUNWsprot SUNWtoo
system SUNWarc     Archive Libraries
system SUNWbtool   CCS tools bundled with SunOS
system SUNWcsu     Core Solaris, (Usr)
system SUNWlibm    Sun WorkShop Bundled libm
system SUNWsprot   Solaris Bundled tools
system SUNWtoo     Programming Tools
```

   If these packages are not available, install them from your Solaris CD-ROM or contact a Sun sales representative.

2. **Obtain the SLK tar file** `lib.JSIIep.tar` **from** `http://www.sun.com/javasystems/slk`**.**

   Instructions for downloading the SLK tar file are provided on the SLK download page.

3. **Untar** `lib.JSIIep.tar` **in a new working directory.**

   For example:

```
% mkdir work
% cd work
% cp download_directory/lib.JSIIep.tar .
% tar xvf lib.JSIIep.tar
```

4. **Create the directories** `classes` **and** `lib` **under the directory** `work/lib.JSIIep`.

```
% cd work/lib.JSIIep
% mkdir classes lib
```

5. **Copy your application class files and other required resources to the `classes` and `lib` directories.**

   Copy all `CLASSPATH`-loadable files to the `classes` directory and all non-class files to the `lib` directory, or create symbolic links to the application build area.

   For example:

```
% cp -r CLASSPATH_loadable_files classes/.
% cp -r non_class_files lib/.
```

6. **Create a file called `javaos.properties` in the `lib` directory that contains JavaOS properties and other properties to be set before the application executes.**

   At minimum, this file should include the property `javaos.mainProgram`, which points to the class containing the `main` entry point of your application. This property is set as follows:

```
javaos.mainProgram=main_class
```

   For example:

```
javaos.mainProgram=Orion.main
```

   For a description of all JavaOS properties, refer to the *Netra j 2.0 Administrator's Guide*.

- **Use `link_javaos` to link your application to the JavaOS binary.**

  Usage for `link_javaos` is as follows.

```
./link_javaos [-sun | -gnu]
```

  where:

  - `-sun` (default) uses the Solaris assembler/linker tools found in `/usr/ccs/bin`
  - `-gnu` uses GNU tools, which must be installed in `/opt/gnu`

  The linking takes between 5 and 20 minutes, depending on the size of your custom application and the speed of your machine.

The `link_javaos` utility does several verifications and then builds a new `javaos` binary that incorporates your application files. The table below shows error messages that may be displayed during each step in the process.

TABLE 3 Error Messages During the `link_javaos` Process

`javaos_link` **action**	**Error Messages**
Verifies your host system is running Solaris 2.5 or 2.6	If your system is running a different version of Solaris, the following message is displayed. `Error: Building "JavaOS" requires Solaris 2.5 or 2.6`

Verifies that the specified tools have been properly installed	If `-sun` was selected (or left default) and the required tools directory is not found, the following message is displayed:
	`Error: The required tools directory (/usr/ccs) was not found`
	If `-gnu` was selected and the GNU tools directory is not found, the following message is displayed:
	`Error: The required tools directory (/opt/gnu) was not found`
	The GNU tools can be obtained from `www.gnu.org`. The minimum GNU packages required are `binutils`, `make`, and `gcc`.
	Note that the SLK build scripts will always look in `/opt/gnu` for the GNU tools. Some systems have the GNU compiler and tools installed in the directory `/usr/local`. On such systems it may be sufficient to add a symbolic link that points `/opt/gnu` to `/usr/local` (e.g. `ln -s /usr/local /opt/gnu`).

Verifies the available Java™ compiler is version 1.1 or later	If not, the following message is displayed. `Error: Java version 1.1 or greater is required`
Verifies the file `javaos.properties` exists	If not found, the following warning is displayed: `Warning: ’./lib/javaos.properties’ was not found.` This file is required to specify the `javaos.mainProgram` property.
Builds a new `javaos`, linking your application files to the binary	During the build, various warning messages may be displayed concerning “impure constant pools,” “references to dead strings,” etc. These messages can be safely ignored.

Reports whether the build is successful	If the build is successful, the following message is displayed:
	`(9) ... 'javaos' successfully created`
	If the build fails, the file javaos is not created and the following message is displayed:
	`Error: 'javaos' was not properly created due to previous errors`

A successful `link_javaos` build yields a single compressed `javaos` file that incorporates your application.

You can now set up `javaos` to be delivered to JavaStations during the boot sequence as described in the *Netra j 2.0 Administrator's Guide*.

Adding an Application Icon to the HotJava Views Selector

Icons in the HotJava Views Selector are part of a HotJava Views group's configuration. To add an icon to a group's configuration, follow the procedure below. See the HotJava Views Administration online help for additional information.

Note – You should specify the server parameters before adding an icon to HotJava Views Selector.

▼ To Add an Application Icon to the HotJava Views Selector

1. **Choose Network Computer Server -> HotJava Views Administration.**

   The HotJava Views Administration page is displayed.

2. **Choose Client-side Configuration -> Groups and Configuration.**

3. **Select the Group to which you want to add the application icon and click the Selector Applications button.**

4. **Choose the application entry in the Application Palette and click Add.**

5. **Click OK.**

6. **Click Set Default Group to set this group as the default.**

   The default is Basic.

   HotJava Views Selector is now configured with the icon.

GO-Joe With RapidX

The Netra j 2.0.1 product contains version 2.01a2 of the GO-Joe X-hosting software. The GO-Joe client-server solution enables access to the Solaris environment and applications from a JavaStation computer.

This section supersedes the following sections in the *Netra j 2.0 Administrator's Guide*

- Using Netra j Connectivity Software, *GO-Joe With RapidX*
- Troubleshooting, *GO-Joe*

GO-Joe is an X server that provides access to all Solaris and X applications within a JavaStation environment.

After configuring the GO-Joe X server, use the Netra j Remote Windowing Tools module to specify the host name or IP address to the Netra server. See “To Reference Remote Windowing Servers” on page 33.

GO-Joe consists of two modules that work together to provide X connectivity in the JavaStation environment: the X-to-RapidX converter and the GO-Joe applet. The X-to-RapidX converter (the GlobalHost ddx loadable module) is installed on a UNIX host that provides connectivity into the X environment on the network. The Java applet enables the user to access Solaris and X Window applications.

System Requirements

You can install and run GO-Joe on SPARC™ systems running the Sun Solaris operating environment, version 2.5.1 or later. The adaptive RapidX protocol and the Java applet total under 500 kilobytes of memory.

Installation

The `netra_install` script does not install the `SUNWgjvxs` server software package. Use `admintool` or `pkgadd` to install this package.

Dependencies

GO-Joe requires additional Solaris software to run properly. TABLE 4 lists the Solaris dependencies.

TABLE 4 Solaris Dependencies for GO-Joe

Package ID	Description	Location
`SUNWcsr`	Core Solaris, (root)	Netra j 2.0.1 software (Solaris 2.5.1 add-on cluster), and Solaris 2.6 CD-ROM
`SUNWcsu`	Core Solaris, (usr)	Netra j 2.0.1 software (Solaris 2.5.1 add-on cluster), and Solaris 2.6 CD-ROM
`SUNWcar`	Core Architecture, (root)	Solaris 2.5.1 or 2.6 CD-ROM

TABLE 4 Solaris Dependencies for GO-Joe *(Continued)*

Package ID	Description	Location
SUNWkvm	Core Architecture, (usr)	Solaris 2.5.1 or 2.6 CD-ROM
SUNWlibms	Solaris bundled shared libm	Solaris 2.5.1 or 2.6 CD-ROM
SUNWtltk	ToolTalk runtime	Solaris 2.5.1 or 2.6 CD-ROM
SUNWxwplt	X Window platform software	Solaris 2.5.1 or 2.6 CD-ROM
SUNWolrte	OPEN LOOK toolkits runtime environment	Solaris 2.5.1 or 2.6 CD-ROM
SUNWoldte	OPEN LOOK desktop environment	Solaris 2.5.1 or 2.6 CD-ROM
SUNWxwdv	X Window system kernel drivers	Solaris 2.5.1 or 2.6 CD-ROM
SUNWxwfnt	X Window system fonts	Solaris 2.5.1 or 2.6 CD-ROM
SUNWdtcor	Solaris desktop `/usr/dt` file system anchor	Solaris 2.5.1 or 2.6 CD-ROM
SUNWmfrun	Motif runtime kit	Solaris 2.5.1 or 2.6 CD-ROM

GO-Joe uses JDK™ 1.1 application program interfaces (APIs), including AWT 1.1 (abstract windowing toolkit) and the New Event Model. Therefore, GO-Joe requires a fully JDK 1.1-capable browser. As of this writing, HotJava Browser 1.1 for the Solaris operating environment is the only browser that supports AWT 1.1 and New Event Model features in JDK 1.1. Netscape Navigator™ 4.03 supports only a subset of JDK 1.1. If you prefer a Netscape browser, check the Netscape web site at `http://devedge.netscape.com/software/index.html` for information about a JDK 1.1 software patch that provides the following functionality:

- AWT 1.1
- New Event Model
- JavaBeans™ Support
- Printing Support for Applets

Note – You must install Netscape Communicator 4.0 and Netscape's JDK 1.1 software patch before running GO-Joe.

Components and Session Overview

A GO-Joe session involves several components working together to bring X Windows to network computer desktops:

- The network computer client(s)
- The X-to-RapidX converter (or RapidX server)
- A Web server to deliver the GO-Joe applet

The RapidX server and the web server could conceivably reside on two different machines. However, Netra j automatically configures them both on the same machine. Netra j also provides an example X session. The GO-Joe session proceeds as follows:

1. You enter a URL in the client system's browser or Java environment loading the HTML page containing the GO-Joe applet from the Web server.
2. The GO-Joe applet prompts you for a login and password to authenticate you to the RapidX server.
3. When you click the Start X Session button (or press Return, depending on the browser), the login and password are sent to the RapidX server at the port specified in the HTML page loaded in step 1.
4. The RapidX server accepts the applet's connection and passes it off to the `go-login` program.
5. The `go-login` program receives the user name and password specified in Step 3. If the user name and / or password are incorrect, an error is returned by the applet and it returns to Step 3.
6. If the user name and password are valid, the `go-login` program starts an X session with special arguments initiating its use of the GlobalHost loadable ddx module.
7. Initialization scripts check for a GO-Joe token, and if present, modify the session startup behavior as specified by the controls in the token parameter.

After these steps, the session is displayed within the GO-Joe applet and you are able to run X clients from the UNIX network.

Web Server Integration

The GO-Joe applet class files (from the `SUNWgjvxv` package) are installed by default into `/opt/SUNWgjvxv`. These files must be accessible to the Java device via the `http` server.

You can make these files viewable by the `http` server by creating a symbolic link in the web server's document root pointing to the installation directory of the GO-Joe applet class files:

```
# cd /var/http/demo/public
# ln -s /opt/SUNWgjvxv GO-Joe
```

This makes the GO-Joe class files accessible with a URL resembling the following: `http://`webserver`/GO-Joe/`.

You should modify the example `xsession.html` file and use it to load the GO-Joe session.

Configuration

The GO-Joe applet ships with an example startup HTML file, `xsession.html`. You should save this file and not modify it. Make a copy of this file and rename it to `x.html`, for example.

The sample `x.html` file contains an example startup session that exercises some of the available parameters for the applet, but this file must be customized before it can be used.

TABLE 5 Applet Parameters

Parameter	Description
width	This parameter is specified in the APPLET tag and determines the width of the GO-Joe frame in the HTML file. Its format is browser-specific, but it can generally be an absolute number of pixels (for example, "width=800") or a percentage of the browser window's width (for example, "width=100%").
height	This parameter is specified in the APPLET tag and determines the height of the GO-Joe frame in the HTML file. Its format is browser-specific, and it can generally be an absolute number of pixels (for example, "height=600") or a percentage of the browser window's height (for example, "height=90%").

TABLE 5 Applet Parameters *(Continued) (Continued)*

Parameter	Description
server	This parameter specifies the name or IP address of the host machine that runs the X session.
port	This parameter specifies the port number that is contacted by the GO-Joe applet. Usually, this is port 491; however, it may be different if the port is being used by a service other than the `go-login` service.
token	This parameter specifies an optional "token" value to be passed into the environment of the X session. See "Advanced Configuration Options" in this chapter for information on the token parameter.

The following is a sample `xsession.html` file that initiates a default session. This sample does not demonstrate all of the available parameters.

```
<HTML>
<HEAD><TITLE>GO-Joe Example Session</TITLE></HEAD>
<BODY>
<HR>
<APPLET ARCHIVE="gojoe.jar" CODE="gojoe.class" WIDTH="800"
HEIGHT="600">
    <PARAM NAME="server" VALUE="myhost">
    <PARAM NAME="port" VALUE="491">
</APPLET>
</BODY>
</HTML>
```

The GlobalHost Loadable ddx Module

The GO-Joe applet communicates through a dynamically loaded Xsun device driver. This driver appears to the Xsun server to be a standard display driver, but transmits display operations to the GO-Joe applet. This is referred to as the GlobalHost loadable ddx module.

To start the Xsun server with the GlobalHost loadable ddx module, you need to reference the device files created in `/devices` similar to the following:

```
/usr/openwin/bin/goinit /usr/dt/bin/Xsession --
/usr/openwin/bin/Xsun :1  -dev /dev/fbs/goglobal0 -I -inetd
```

When the Xsun queries this device, it loads the correct module for GO-Joe.

Note that you would never type this command at the command line. It is started by a script after the `go-login` program has authenticated the user through `inetd` or started by the `go-login` program itself.

The `go-login` Authentication Program

The `go-login` program is started by `inetd` and receives user authentication information from the applet before starting the session. The `go-login` program is designed to be called directly from `inetd`. Some implementations of `inetd` software limit the number of command-line arguments that can be passed to the `go-login` program, making it necessary to call the `go-login` program from a goshim script. The goshim ensures that all of the arguments are passed to the `go-login` program. To use a goshim, `go-login` must be installed into your `inetd.conf` file as follows:

```
go-login stream tcp nowait root /usr/openwin/server/etc/goshim
goshim
```

Adding the GO-Joe Icon to the HotJava Views Selector

Icons in the HotJava Views Selector are part of a HotJava Views group's configuration. To add the GO-Joe icon to a group's configuration, follow the procedure below. See the HotJava Views Administration online help for additional information.

Note – You should specify the server parameters before adding an icon to HotJava Views Selector.

▼ To Add the GO-Joe Icon to the HotJava Views Selector

1. **Choose Network Computer Server -> HotJava Views Administration.**

   The HotJava Views Administration page is displayed.

2. **Choose Client-side Configuration -> Groups and Configuration.**

3. **Select the Group to which you want to add the GO-Joe icon and click the Selector Applications button.**
4. **Select the GO-Joe entry in the Application Palette and click Add.**
5. **Click OK.**
6. **Click Set Default Group to set this group as the default.**

   The default is Basic.

   HotJava Views Selector is now configured with the GO-Joe icon.

Running GO-Joe

To start a GO-Joe session, load the HTML file with the GO-Joe applet into your Java environment. GO-Joe prompts you for a user name and password for the session, and after authentication, provides the X display on your Java desktop.

This session is almost identical to running an X session on the system console, with a few differences, which are described below.

DISPLAY Environment Variable

The `$DISPLAY` environment variable tells X clients where to contact your X server. GO-Joe sets this variable to point to an alternate display on your host machine. For example, if your UNIX host is named `workstation` and the Java device is named `java`, you might expect the `$DISPLAY` variable to set `java:0` as its value. However, GO-Joe uses the host name for its `$DISPLAY` variable value, in this case, `workstation:1`.

GO-Joe makes an additional optimization that can be somewhat confusing. In X parlance, if the `$DISPLAY` variable is set to `unix:#`, the JavaStation client attempts to connect using a local transport. For example, instead of using TCP/IP, it connects using a named pipe. This connection is faster than using TCP/IP for clients running on the same host. However, the `unix:#` value cannot be used if you run clients from different hosts.

Instead, use the following shell script to change the `$DISPLAY` variable to point to the host name of your machine. This translates `$DISPLAY` from `unix:3`, for example, to `workstation:3` enabling X clients on other machines to successfully contact GO-Joe on the workstation host. This can be included in the `.profile` file:

```
DISPLAY=`/bin/uname -n``/bin/expr $DISPLAY : '[^:]*\(:.*\)'`
```

Using a Two-Button Mouse

X requires and assumes the availability of a three-button mouse. Most Java environments provide a two-button mouse. In an X session, the left and right mouse buttons correspond to the left and right mouse buttons under X. GO-Joe maps the simultaneous pressing of both mouse buttons into a middle mouse button press.

The Mouse Arrow

Current AWT implementations provide only limited support for specifying the shape of the mouse arrow in the Java environment. For this reason, GO-Joe currently does not change the shape of the mouse arrow.

Advanced Configuration Options

You can configure GO-Joe to provide a variety of functions. This flexibility can be complex and should only be done by advanced users or system administrators.

Token Parameter

The GO-Joe applet accepts an optional token parameter in its HTML file. If the token parameter is present, the applet transmits it to the `go-login` program, along with the user name and password. The `go-login` program creates an environment variable, `$GG_TOKEN`, which is available for other startup scripts to process.

The following three controls are supported by the token parameter:

- *Session Control* `[session=<openwin|cde>;]`—This control is used to specify the session as either OpenWindows™ or CDE. If this control is not set, OpenWindows is the default desktop on Solaris version 2.5 and earlier, and CDE is the default for Solaris version 2.6 and higher. The session control is optional.
- *Window Manager Control* `[wm=</path/to/window/manager>|nowm>;]`—Alternate window managers can be specified by using this control. For security reasons, you must specify an absolute path corresponding to an executable window manager. Therefore, no arguments can be passed to the window manager. The window manager control is optional.

Note – The X session terminates when the key client terminates. Normally the window manager is the key client and provides a menu item or button to exit the session. If you specify `nowm`, one of the clients started in the case specified by the Startup Control needs to become the key client. Therefore, if `nowm` is specified, a startup control is required.

- *Startup Control*—The startup control must correspond with a `$GG_TOKEN` case in either `$HOME/.gotoken-init` or `/usr/openwin/lib/gotoken-init`.

The Solaris session startup files (OpenWindows or CDE) are initiated when the GlobalHost loadable ddx module is installed.

When starting an OpenWindows session, the token parameter is used in `$OPENWINHOME/lib/gotoken-init` or in `$HOME/.gotoken-init`. If `$HOME/.gotoken-init` exists, it is used before `$OPENWINHOME/lib/gotoken-init`. System administrators and system integrators can create system-wide token processing routines by modifying `$OPENWINHOME/lib/gotoken-init`, while still enabling users to override these settings in the `$HOME/.gotoken-init` file. In addition, if the `$GG_TOKEN` variable is not set, the session startup is the same as a standard OpenWindows session.

The CDE startup mechanism is somewhat different from the OpenWindows mechanism. Unlike OpenWindows, which uses a single `openwin-init` file, CDE stores its session initialization as a directory filled with several files. Because of this, the CDE token is used to specify a session to be started. Sessions are stored in `/usr/dt/config/dtgotokens`, in subdirectories that match the name of the startup control. When the session is started with the CDE token, the session directory is copied into the user's home directory, and CDE starts this session for the user.

Structure of `gotoken-init`

The `gotoken-init` is based on the standard `openwin-init` file, with the addition of the following section:

```
    if [ "$OW_WINDOW_MANAGER" = ':' ]; then
       toolwait=
    else
       toolwait=toolwait
    if
    unset OW_WINDOW_MANAGER
    case "$GG_TOKEN" in
    xterm)
    $toolwait $OPENWINHOME/bin/xterm
    ;;
    *)
        echo>&2"$OPENWINHOME/lib/gotoken-init:error:\'$GG_TOKEN\':
case not found."]
    echo >&2 "    Using defaults."
    . $OPENWINHOME/lib/openwin-init
    ;;
esac
```

This section parses the $GG_TOKEN (with any session or window manager controls removed) and starts the appropriate client or clients. In the example, only one startup control, xterm, is defined. Any other token returns an error and use the default OpenWindows startup.

If nowm is specified, the $toolwait variable sets the $OW_WINDOW_MANAGER variable (in conformance with the standard OpenWindows method of specifying an alternate window manager) to the colon. This results in no window manager being executed when the shell interprets the colon as a null command.

If you modify the $toolwait variable when there is a window manager running, the $toolwait program is invoked to start xterm (which runs in the background). When no window manager is running, the last client started by gotoken-init must not run in the background, or the GlobalInit program thinks that the session is over and shuts down the X server. Conditionally running the toolwait program solves this problem.

Structure of dtgotokens Directory

The dtgotokens directory contains subdirectory names that match the possible values for the startup control in $GG_TOKEN. Each subdirectory contains the files necessary to start a CDE session with the X applications that are appropriate. The example directory, xterm, starts a single xterm window.

To create additional tokens, first run a CDE session and start the appropriate clients and applications. Second, exit the CDE session. The session is saved in ~/.dt/ sessions. Copy the session files into the dtgotokens directory using a command similar to the following:

```
# cp -R ~/.dt/sessions /usr/dt/config/dtgotokens/sampletoken
```

The -R argument is used to copy recursively (including subdirectories). You do not have to create the sampletoken directory before executing this command.

GO-Joe Diagnostics and Troubleshooting

GO-Joe provides diagnostic tools and outputs to help diagnose problems that may arise due to misconfiguration and other difficulties. Check these if you encounter any problems.

JavaOS Console Output

On devices that provide it, the JavaOS console output (see “Troubleshooting”) can be highly informative if the applet terminates prematurely. When viewing this output, look for all exceptions that may be listed. Exceptions may occur that cause further exceptions as the program continues to execute, and it is usually the original exception that indicates the true cause of the problem. In addition, the GO-Joe applet prints messages to the status bar, but not all Java environments show this status (or they may overwrite it with their own status messages). These messages are also sent to the Java console, so they may be visible in the console log when they are not visible on the status line.

The `/tmp/Xerr:n` Error File

The GlobalHost loadable ddx module redirects the standard error output for the session to a file called `/tmp/Xerr:n`, where *n* represents the display number of the session. This file contains diagnostic messages from the `GlobalInit` program, from the ddx loadable module itself, and from the X clients that run throughout the session.

Common Problems

The GO-Joe product has been designed to be easy to configure and use, yet is somewhat complex in operation. Some of the more common problems encountered are described here.

HTML References

If the GO-Joe applet fails to load entirely, check the HTML file you are loading and verify that the `APPLET` tag is correctly formed. Check the codebase path and the path and file name to the applet file itself. Finally, investigate the log files for your HTTP server (usually called `access_log` and `error_log`) to see if the applet is being successfully transmitted to the Java environment. It may help to exit your browser or Java environment and restart it to clear any cached files that may be interfering with the applet's execution.

Java Security Manager Exceptions

All Java environments implement a security manager that determines what operations may be dangerous for an applet to perform and can enable or restrict these actions as it sees fit. The biggest restriction that most Security Managers

implement with respect to GO-Joe is that an applet is allowed to connect only to the host that served that applet. If you have such a Security Manager, your GO-Joe applet is only able to connect to the `go-login` program running on the server that the applet was loaded from. In addition, the `diagfile` functionality of GO-Joe is similarly restricted.

The JavaStation produces a message as follows:

```
Sunw.hotjava.applet.AppletSecurityException:
checkconnect.networknone
at sunw.hotjava.security.CommonSecurity.checkconnect()
at java.net.InetAddress.getAllByName)()
```

Remote Windowing Procedures

Configure the Netra j server by using the Netra j Remote Windowing Tools module to reference remote windowing servers.

▼ To Reference Remote Windowing Servers

Note – Remote Windowing servers must be accessible to the Netra j server.

---

1. **From the Main Administration page, under “Remote Windowing Administration,” click Remote Windowing Tools.**

   The Remote Windowing Administration page is displayed.

2. **Select one of the following:**

   - To reference a Citrix WinFrame server, click Configure Citrix.
   - To reference a GO-Joe X server, click GO-Joe.

3. **Complete the form using the information in**TABLE 6**.**

TABLE 6 Remote Windowing Information

Item	Description
Citrix Server Host Name or IP Address	The host name or IP address of the server running WinFrame.
Go-Joe Server Host Name or IP Address	The host name or IP address of the server running GlobalHost module.
Go-Joe Port Number	The port number in which the GlobalHost server (X server) is running. The default port is 491.

Netra j 2.0 Documentation Updates

This section describes updates to the *Netra j 2.0 Installation Guide* and the *Netra j 2.0 Administrator's Guide*. Both documents support Netra j 2.0.1.

Netra j 2.0 Installation Guide

- Under “Initial Configuration” on page 18, you are no longer required to complete steps 3 through 9. The Netra j 2.0.1 software will not display these configuration pages.
- Under “Installation Notes” on page 10, the note entitled “Brick Model Clients” should be removed.

Netra j 2.0 Administrator’s Guide

This section describes updates to the *Netra j 2.0 Administrator’s Guide*.

DNS Primary Server Administration

- TABLE 7 DNS Primary Server Administration should be updated with the following information:

TABLE 7 DNS Primary Server Administration

DNS Primary Server Information	Description
Primary Domain Name	The name of the primary domain. The primary domain name is assumed to be fully qualified (do not use a trailing period). Example: `elmo 129.144.79.1`
Domains / DNS Servers	The domains served by the corresponding DNS name servers. The domain must be a subdomain of the primary domain, unless it is a fully qualified domain name. The host name of the name server must be within this domain, unless it is a fully qualified host name. All DNS servers are treated as authoritative for the listed domain. If a subdomain is entered, authority for that domain is delegated to the listed server. This field must always contain a record for this name server. Enter the name of this primary domain together with the host name of this server. A trailing period is required for the domain name.

- The instructions under “To Modify or Delete a DNS Primary Domain” should state that the DNS Administration page appears, not the DNS Primary Domain page.

DNS Primary Domains

The domain name should be followed by a trailing dot as shown in the following example:

```
mydomain.com.
```

Adding Records for Name Servers

When adding records for name servers, the first record must contain at least one entry specifying that this server is a name server for the domain. You must also create a second record for the name server of the subdomain within the primary domain.

Example of DNS Primary Server

The example of the DNS Primary Server under “Name Services Administration,” “DNS” should be replaced with the following information:

For example, assume that a master database for the domain `comedy.tv.net` is to be set up. In the Primary Domain Name field, specify:

```
comedy.tv.net
```

In the Host Names / Host Addresses field, type a list of those hosts whose presence are to be broadcast to any machine that can connect to this DNS server. For this example, type `ren` and `stimpy`, and for a host called `homer` located in the DNS subdomain `black.comedy.tv.net`, type `homer.black`.

```
ren             129.1.1.2
stimpy          129.1.1.3
homer.black     129.2.1.2
```

The Internet community uses conventional names for hosts that provide certain types of services, in order to make them easy to locate. For instance, the WWW server for a domain is usually known as `www.domain`, and an anonymous FTP server is typically called `ftp.domain`. On `comedy.tv.net`, `ren` is an FTP and WWW server, while `stimpy` is a name server. Standard aliases for these machines are added into the Host Aliases / Host Names field. For example:

```
www             ren
ftp             ren
ns              stimpy
```

`stimpy` is going to handle mail sent to `comedy.tv.net`, so an MX record needs to be created:

```
comedy.tv.net. 5   stimpy
```

Finally, records are added for name servers. This field must contain at least one entry specifying that this server is a name server for the domain. Add a second record `homer.black`, which is the name server for a subdomain of this domain. If a domain contains subdomains, the DNS primary domain server for the domain must know a DNS server for the subdomain. `comedy.tv.net` has a subdomain called `black`, whose DNS primary domain server is called `homer`. `homer` resides on that

domain, so in order for the Netra DNS server to reach `homer` when it needs to request name services for that domain, it has to know `homer`'s IP address. That is why `homer.black` was added to the list of known hosts.

Modem Administration

Connection speed choices shown in the table "Modem Remote Host Administration" include 38400, 19200, and 9600. Actual choices available in the Netra j "Add a Remote Host Connection" page are as follows:

- 9600
- 19200
- 38400
- 57600
- 76800
- 115200
- 153600
- 230400
- 307200
- 460800

Default Web Server Administration

The section "Default Web Server Administration" under "Using Netra j System Administration" (Chapter 6) should be removed.

File System Backup and Restore

The tables that describe options on the following pages do not include a complete description of the "Other directories" text field:

- File System Backup Options
- Easy File System Restore
- Selective File System Restore

The "Other directories" text field can be used to specify another top-level directory to be saved in the backup. It also *must* be used to specify an alternate web server document root (i.e. other than `/export/htdocs`) if HTML documents will be saved in the backup.

Save and Restore Configuration

The instructions under "To Eject a Diskette" should state that the Verify page, not the Eject Diskette page, appears after you click Eject Diskette.

Routing Administration

In TABLE 4-6 Dynamic Router Administration and TABLE 4-7 Static Router Administration, the Destination Network/Host Address option should be Destination Address.

Network Computer Server Administration

The table under “Configuring Global Parameters” states that Time Zone is set in a text field, when actually it is set using a scrollable menu. Since a menu is used, an invalid entry is not possible.

Setting JavaOS Properties

The name of the JavaStation serial port is now `SerialA`. This change affects the setting of the JavaOS property `javaos.printservice.local.params.serial-`*port* described under “Printing Properties.”

TABLE 8 Printing Properties

Property Name	Default Value	Description
javaos. printservice. local.params. serial-*port*	null	The *port* portion of this property is the name of a serial port (which is always SerialA on the JavaStation). This property sets the communications parameters for the serial port. The syntax of the parameters is *baud_rate*:*data_bits*:*stop_bit*: *parity*: *flow_control*. For example: `-djavaos.printservice.local.params.serial-SerialA=57600:8:1:none:hh` Valid values for each parameter are as follows: • *baud_rate*: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 • *data_bits*: 5, 6, 7, 8 • *stop_bits*: 1, 1.5, 2 • *parity*: none, odd, even • *flow_control*: none, {s,h}{s,h} You can disable flow control by specifying none. Otherwise, specify an {input}{output} pair by selecting from Xon/Xoff (s) or RTS/CTS (h) for input and output.

The description of the property `javaos.homedir` under “General Properties” should be changedto the following:

TABLE 9 General Properties

Property Name	Default Value	Description
`javaos.homedir`	null	This property specifies the NFS path JavaOS should mount if NIS is not used to find the path based on the user name. The NFS path is specified as `hostname:/path`. This property is most often used to determine the directory to use for the `properties` file that is read by HotJava at startup. If the `javaos.login` property is set to false, `javaos.homedir` is not used.

JavaStation User Setup Forms

Information for PPP/modem described in the JavaStation User Setup forms available in Appendix C and from the Netra j Administration Interface should be updated with the following information:

- Step 4 should be removed.
- In the Script Options window, the phrase “Connect to host” should supersede the phrase “Connect to a command line host.”

Localization and Fonts

Under “Modifying Languages Displayed at Login,” the property setting examples should be changed to the following.

```
-djavaos.loginLocaleList=locale-1;locale-2; ... ;locale-n
```

The default value of this property is:

```
-djavaos.loginLocaleList=en_US;de;es;fr;it;ja;ko;sv;zh;zh_TW
```

Troubleshooting

The following sections pertain to the “Troubleshooting” appendix in the *Netra j 2.0 Administrator’s Guide*.

Delete Tables

The following tables should be deleted.

- TABLE D-2 Boot Problems and Suggested Actions
- TABLE D-5 Additional Error Messages and Known Problems

Console Error Message

If you configure the Netra j system as a NIS master server or slave server, or if you send mail from the Netra j system, the following message appears on the console:

```
sendmail[16702]: NOQUEUE: SYSERR(root): Cannot bind to domain
netratest.Eng.Sun.COM: no such map in server's domain: Bad file
number
```

This message can safely be ignored.

Troubleshooting JavaOS

The following section, "Capturing Log Files," should be added to the *Troubleshooting* appendix.

Capturing Log Files

In some cases, if JavaOS fails, it may broadcast an SNMP trap. The trap can be received by any SNMP manager listening on the (sub)net. There is a simple SNMP trap receiver supplied with JavaOS state information in a log file.

▼ To Capture Log Files for JavaOS

- **Run the following command on any machine in the (sub)net:**

```
# /opt/SUNWjsos/bin/snmptrapd -x /opt/SUNWjsos/bin/logdumper&
```

If failures do occur, they are saved in the `/tmp` directory of the server machine with the following unique file name:

`/tmp/javaos.log.`*IP address of failed client@time in seconds since January 1, 1970*

You can browse the log file to determine errors that may have caused a failure. You may also want to add the `snmptrapd` command to the Solaris initialization and booting hierarchy. See the `README` files in `/etc/init.d` and `/etc/rc2.d` for details.

Note – Not all JavaStation client failures will result in a log file creation on the server machine.
