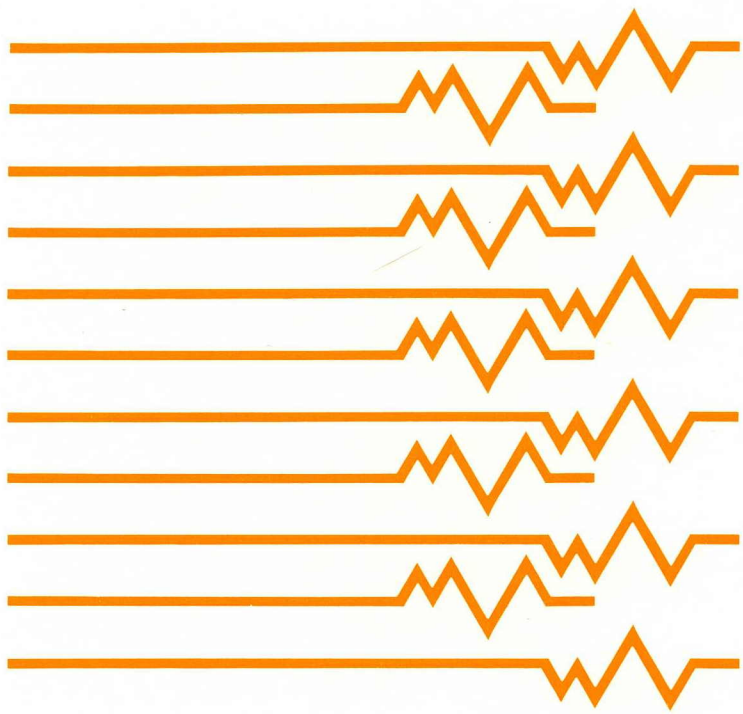
XEROX

Xerox GLOBALVIEW

Document Services for Sun



Technical Reference Manual

XEROX

GLOBALVIEW Document Services for Sun

Technical Reference Manual

June 1992

Xerox Corporation
Product Education and Documentation
PAHV-123
3400 Hillview Drive
Palo Alto, California 94304-1300

Rank Xerox Ltd.
Multinational Customer & Service Education
Welwyn Garden City
Hertfordshire AL7 1HE
England

©1992 by Xerox Corporation. ©1991 by Siemens Nixdorf Informationssysteme AG
All rights reserved.
Published June, 1992.

Copyright protection claimed includes all forms and matters of copyrightable material and information now allowed by statutory or judicial law or hereafter granted, including without limitation, material generated from the software programs which are displayed on the screen such as icons, screen displays, looks, etc.

Publication number: 610E26750

Xerox®, GLOBALVIEW®, and all Xerox product names mentioned in this publication are trademarks of Xerox Corporation.
PostScript is a trademark of Adobe Systems Inc.
HP®, HP LaserJet III, and HP LaserJet IIISi are trademarks of Hewlett Packard Company.
Rank® Xerox is a trademark of Rank Xerox Ltd.
AT&T® is a trademark of American Telephone & Telegraph Company in the U.S.A. and other countries.
SparcPrinter®, Sun®, SunInstall®, SunOS®, SunView®, SunWindows®, and X11/NeWS® are trademarks of Sun Microsystems, Incorporated.
UNIX® is a trademark of AT&T.

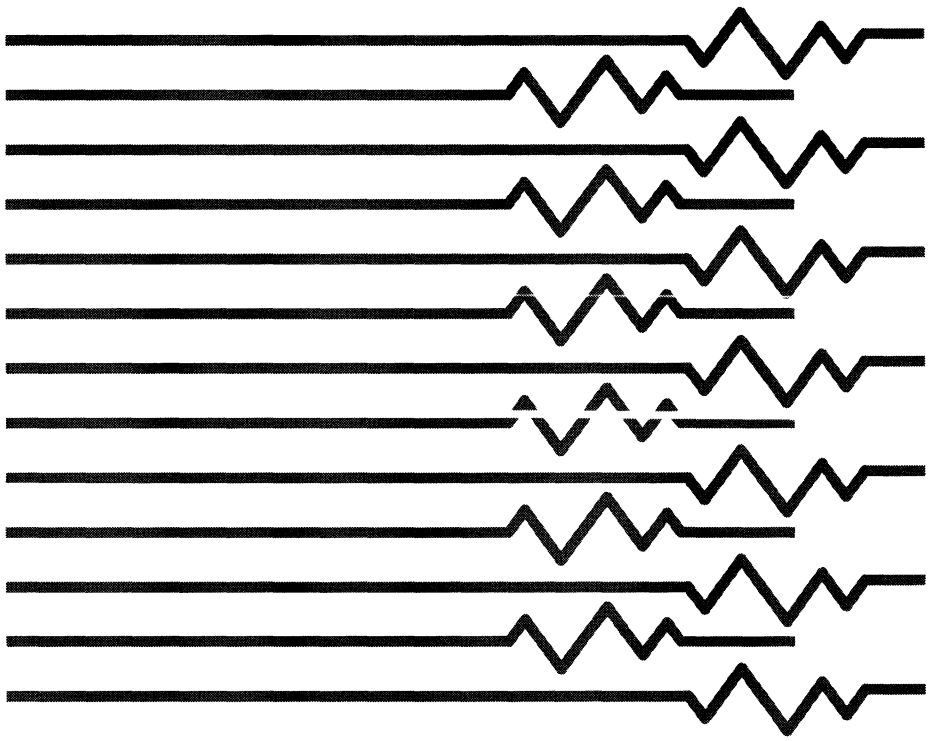
Not all the products mentioned in this publication may be available in your country. Please contact your local representative for details.

Changes are periodically made to this document. Changes, technical inaccuracies, and typographical errors will be corrected in subsequent editions.

The body of this document was produced on a DocuTech Publishing System.

Table of contents

Overview



1. Introduction

The *GLOBALVIEW Document Services for Sun Technical Reference Manual* provides in-depth information about network service operations, and describes the two GLOBALVIEW Document Services (GVDS) user interfaces:

- The Service Executive interface is window-oriented, and is used to manage the Clearinghouse, Mail, and Print services.
- The TTY interface is command line-oriented, and is used to manage the File Service.

This chapter explains how to use this technical reference, briefly describes the volumes in the GVDS Network Administration Library, and shows the type conventions we use to help you recognize information.

How to use this Technical Reference Manual

Some of the basic GLOBALVIEW Document Services (Services Environment, Clearinghouse Service and Mail Service) have been part of Xerox Network Services for several years; although their functionality has been enhanced or modified with successive releases, the basic functionality is pretty stable. Other services (File Service and Print Service) are significantly different in this software release.

Rather than writing one all-inclusive chapter for each of the GLOBALVIEW Document Services, we have provided as much relevant information as possible for each individual service. Sometimes the information can be handled in a single chapter; sometimes, it requires several chapters to be as complete as possible.

To make it easier to navigate through the service-oriented material, we divide this technical reference into *sections*. Each section may contain one or more chapters on the related service.

Overview

The Overview contains general information about this technical reference, and about the overall structure of the GVDS environment.

Chapter 1, Introduction explains how to use this technical reference and describes the volumes in the GVDS Network Administration Library.

Chapter 2, User Interface, describes GVDS's dual user interface: a GUI for the Service Environment, Clearinghouse Service, Mail Service, and Print Service, and a TTY interface for the File Service.

Clearinghouse Service

The Clearinghouse Service section contains technical information related to the CHS.

Chapter 3, Clearinghouse Service overview, provides an overview of CHS operation, including Clearinghouse components and service design.

Chapter 4, Clearinghouse protocols, describes standard protocols that perform fundamental roles in the Clearinghouse architecture, and the data format standard that defines the structure of the entries stored in the Clearinghouse database.

Chapter 5, Clearinghouse remote errors, describes what happens when Clearinghouse and Authentication remote procedures fail.

Chapter 6, Clearinghouse references, lists Xerox documents used to create the Clearinghouse Service section.

Mail Service

The Mail Service section contains technical information related to the Mail Service.

Chapter 7, Mail Service overview, describes the programs which implement the Xerox Mailing Protocols Standard.

Print Service

The Print Service section contains technical information related to the Print Service.

Chapter 8, Print Service overview, describes the programs which implement the Xerox Mailing Protocols Standard.

File Service

The File Service section contains technical information related to the File Service.

Chapter 9, File Service overview, describes the design of the Xerox File Service.

Chapter 10, File Service error and information messages, lists all error messages the File Service may display at the server or at the user's workstation.

Chapter 11, File Service volume repair, describes how the File Service maintains its consistency during each online volume operation.

Chapter 12, NFS restrictions, describes how security is implemented between the File Server and Sun's Network File System (NFS) protocol.

Chapter 13, File Service backup, describes how to use native UNIX utilities (such as dump, tar, and bar) or third-party utilities to back up and restore XNS files.

Chapter 14, UNIX/XNS access, explains how the File Service manages access to files, and how UNIX access permissions are set for stored files.

Chapter 15, File Service executable files and data, shows the location and function of each File Service component file.

Books in the Network Administration Library (NAL)

The *GLOBALVIEW Document Services for Sun Network Administration Library (NAL)* provides information System Administrators need to know in order to set up their network and install the network software. It also provides reference material for maintaining and troubleshooting the network.

In addition to the *Release Notes* (packaged with the GVDS software), the NAL includes the following books:

- *Introduction to GLOBALVIEW Document Services*—Provides an overview of each of the network services provided in GVDS for Sun: Clearinghouse Service, File Service, Mail Service and Print Service.
- *Installation Guide*—Describes how to install and configure GVDS in a network of Sun servers and workstations.
- *System Administration Guide*—Contains guidelines and requirements for planning the network and its resources. It includes a series of System Administration Worksheets the System Administrator can use to set up the network after GVDS software installation is completed.

  (The System Administration Worksheets are also packaged separately, in large size format, with the GVDS software.)
- *System Administration Reference*—Gives in-depth information about network service maintenance, including how to use the graphical user interface to add, change, delete, move, and duplicate services, users, or other network objects.
- *Troubleshooting Guide*—Describes how to isolate a network problem based on observable symptoms and the diagnostic procedures in the server's hardware reference guide.
- *System Messages*—Describes the error messages the user may see on the workstation/server and how to diagnose and correct them.

Documentation conventions

This book uses the following conventions to help you recognize information.

Commands

GLOBALVIEW Document Services for Sun supports two different types of commands. Each has its own type convention.

[Commands in headers] Commands you select with the mouse (such as commands in window headers, option sheet headers, or property sheet headers) are marked with square brackets ([ ]). For example:

Use the Mail Service [Add Mailbox] command ...

typed-in commands Commands which you type in at the command prompt (such as File Service commands or UNIX commands) are shown in **boldface**. For example:

Type **Add File Drawer** ...

UNIX commands are case-sensitive. When you need to use a UNIX command, type it in exactly as it is shown.

Keys to press

Keys on the workstation keyboard are marked with angle brackets (< >). For example:

... and press <Return>.

Variable information

Variable information, such as names and numbers specific to the operation, appears within angle brackets (<>). For example:

Swap space on <hostname> is sufficient.

2. User interfaces

This chapter provides a description of the Service Executive (SE) and TTY user interfaces. These interfaces are the means by which the System Administrator enters commands and displays the results of operations.

The SE interface is supplied by a software component called the Service Environment. The Service Environment is the underlying software which permits three Xerox network services to operate on a network comprised of Sun-based workstations. These services are the Clearinghouse Service, Mail Service, and Print Service.

The TTY interface is supplied by a software component called Services Common Software for UNIX (SCSX). SCSX is the underlying software which permits the File Service to run on a Sun-based Xerox network.

The principal distinction between the two interfaces is the method of command entry used by the System Administrator. The SE interface is window-oriented, like most GLOBALVIEW applications. The TTY interface is command line-oriented, as is the case with the interface for Xerox 8000 and 8090 servers and the UNIX and MS-DOS operating systems.

Both interfaces and both underlying software provide:

- A means of command entry
- A means for the software to send messages to the operator
- A means of accomplishing communications between clients and services
- A set of general server commands.

The primary user of both the SE and TTY interfaces is the System Administrator. However, any user can access the Print Service through the Service Executive interface. In addition, some System Administrators may delegate certain File Service maintenance responsibilities (for example, file drawer access list maintenance) to any user who is a file drawer owner.

This chapter describes the entry techniques and display elements which support user interaction with the system. It does not describe the broader subject of the underlying software. However, there are elements of the underlying software that must be discussed to explain certain interface principles.

Service Executive interface

The Service Executive interface is supplied by software components called SECore and SEExecutive, which are collectively referred to as the Service Environment.

This section discusses the following subjects:

- Purpose
- Graphical User Interface
- On-line help system
- Window management
- Server commands
- Software components
- Command structure
- Normal startup.

Purpose

The Service Executive interface has several major purposes:

- It provides a fast, easy-to-use Graphical User Interface (GUI) for the System Administrator. The System Administrator manages servers and services, using simple commands, from windows which are identical in operation to GLOBALVIEW windows.
- It allows the System Administrator to manage servers and services remotely. Services are managed from the System Administrator's own workstation. Servers and services can also be administered at their physical (local) locations.
- It defines a method of command processing that all services must follow. In programming terms, these commands consist of phrases, predicates, and call-back procedures.
- It supplies a set of server commands used to perform common operations for all services.

Graphical User Interface

This section provides a brief overview of the major components of the Service Executive Graphical User Interface (GUI.) You can find more information about the GUI in the *System Administration Reference*.

Service Executive icon

The System Administrator accesses servers and services through the Service Executive icon. Blank Service Executive icons are available to the System Administrator in the Basic Icons folder after a successful installation of the Service Environment. The System Administrator then names the icons.

Figure 2-1 shows a series of named Service Executive icons on the System Administrator's desktop.

Figure 2-1. **Service Executive icons**

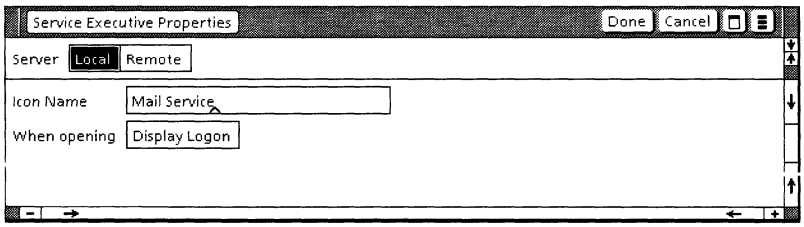
When the System Administrator opens a Service Executive icon, the Service Environment software initiates binding to a server, as specified in the icon's properties sheet. A logon sheet then appears.

Properties sheet

The System Administrator uses the icon's properties sheet to provide the parameters for binding an icon to a server.

Figure 2-2 illustrates a typical Service Executive icon properties sheet.

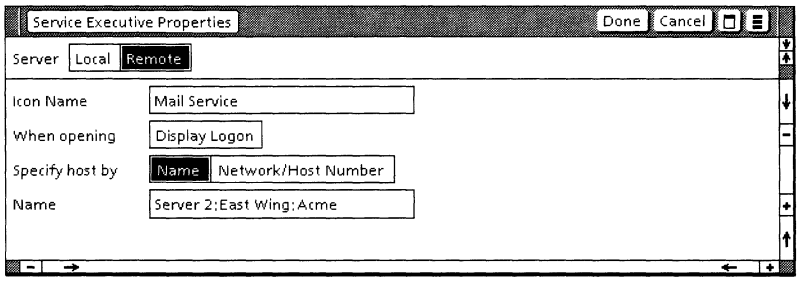
Figure 2-2. **Service Executive icon properties sheet**



In the above example, the System Administrator selected [Local], indicating that the service to be managed is resident on the same workstation or server as the Service Executive icon.

To manage a remote server or service, the System Administrator selects [Remote] and then specifies a remote workstation or server. The remote server can be addressed by its fully-qualified Clearinghouse name or by its network and host numbers. Figure 2-3 illustrates a properties sheet where remote management has been selected.

Figure 2-3. **Selection of remote management**

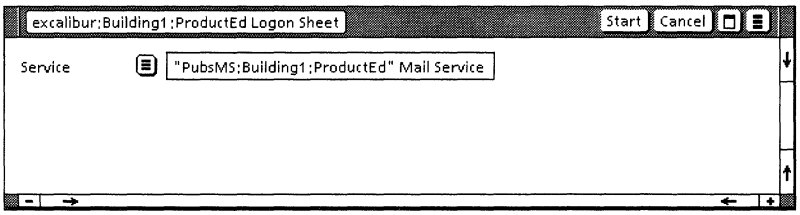


The System Administrator can also choose whether logon information will be required when the icon is opened. If the System Administrator selects [Display Logon], the system requests logon information. If the field is not selected, the system uses the workstation logon information already in effect.

Logon sheet

When the System Administrator opens a Service Executive icon, a logon sheet appears. Figure 2-4 illustrates a typical logon sheet.

Figure 2-4. **Service Executive logon sheet**



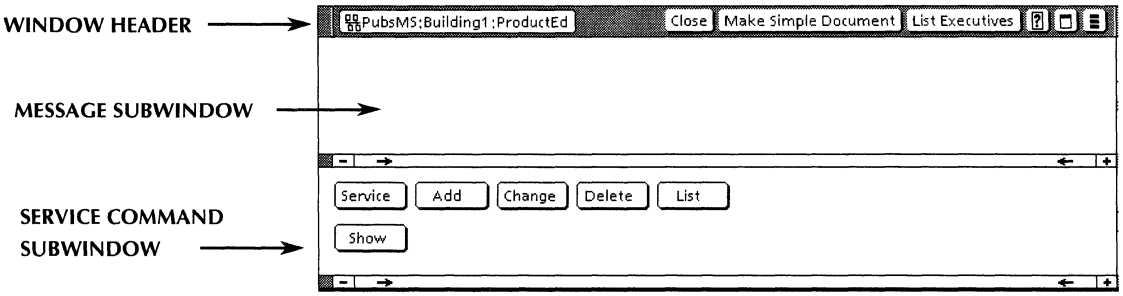
The auxiliary menu in the window permits the System Administrator to select one of the services on the server for management activities.

When the System Administrator selects [Start] in the logon sheet header, the Service Executive software initiates binding to the selected service. A Service Executive window for managing the selected service appears.

Service Executive window

After logon, the Service Executive window appears. Figure 2-5 illustrates the Service Executive window.

Figure 2-5. **Service Executive window**



The Service Executive window has three areas: window header, message subwindow, and service command subwindow.

Window header Contains the service name, commands, and auxiliary menus.

Message subwindow Displays the results of system administration operations.

Service command subwindow Contains "buttons." Buttons are objects which allow the System Administrator to access service commands. The buttons vary depending on the service; however, the Service Environment provides similar commands to the various services. For example, each service has [Service], [Add], [Change], [Delete], [List], and [Show] buttons.

When the Service Executive window appears, it reflects commands for a particular service. Typically the window is referred to by its service name (for example, the Mail Service executive window).

Auxiliary menus

The Service Executive window has three auxiliary menus: Help, Window Management, and Floating Items.

Help Provides access to an on-line help system for server and service commands.

Window management Provides access to window management commands which are consistent with other GLOBALVIEW operations.

Floating items Provides access to server commands.

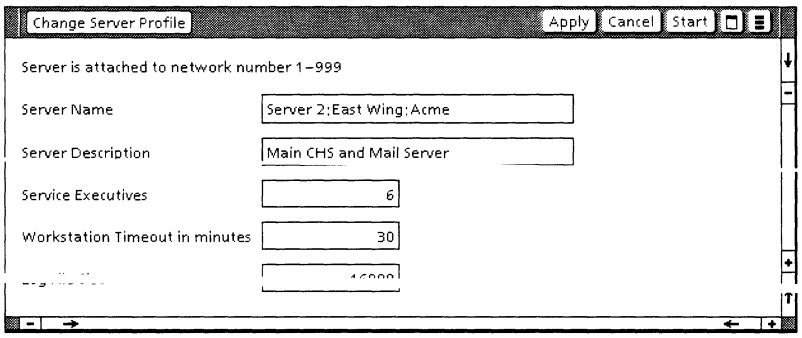
Option sheets and text windows

The GUI uses two other window types: Option sheets and Text windows.

Option sheet Permits the System Administrator to make selections and enter data.

Figure 2-6 illustrates a typical option sheet.

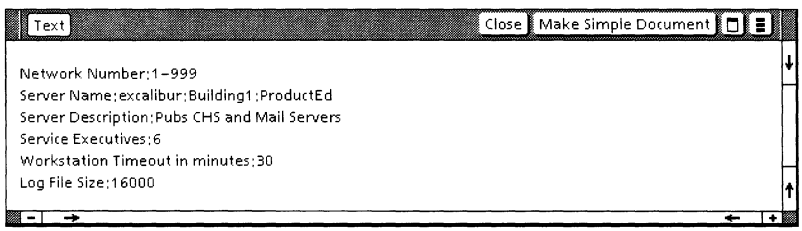
Figure 2-6. **Typical option sheet**



Text window Displays the results of [List] and [Show] commands. The System Administrator can size and move the window, scroll through the text, and make a simple document.

Figure 2-7 illustrates a typical Text window.

Figure 2-7. **Text window**



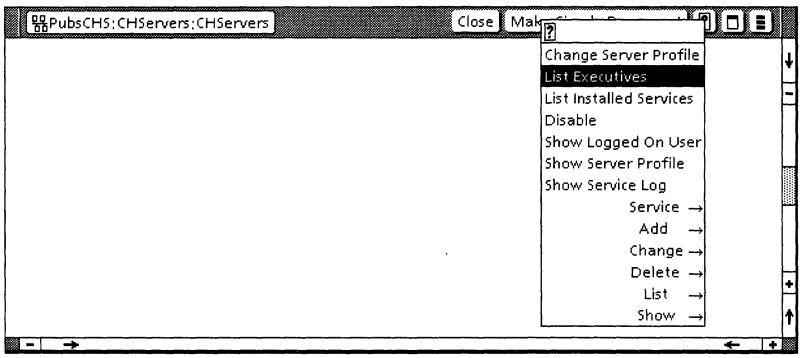
When the System Administrator selects [Close], the window and its contents are not saved.

On-line help system

The on-line help system contains brief explanations of commands. The System Administrator accesses the help system by selecting an item from the Help auxiliary menu.

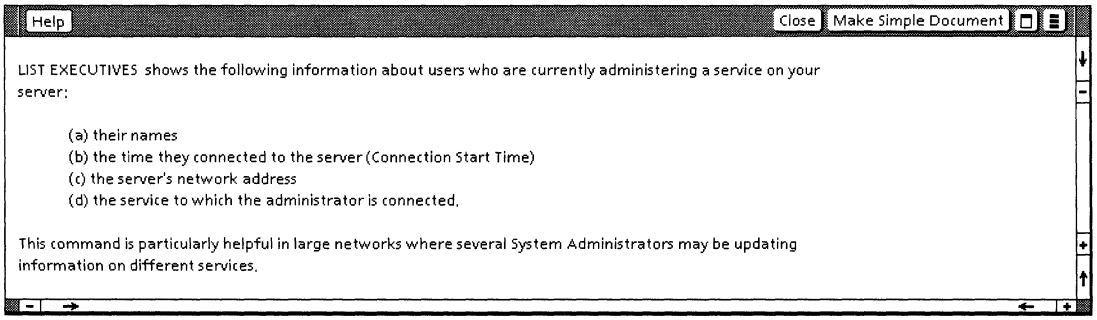
Figure 2-8 illustrates accessing the help system.

Figure 2-8. **Help system access**



After selecting a command from the Help auxiliary menu, the Text window displays a brief description of the command. Figure 2-9 illustrates a Help Text window.

Figure 2-9. **Help Text window**

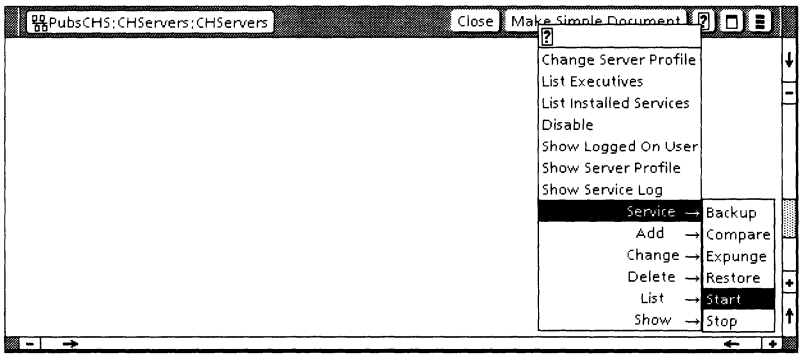


Cascading menus

Help selections for service commands have subordinate selections, which appear in cascading menus.

The Help auxiliary menu displays an arrow (→) next to selections which have cascading menus. Figure 2-10 shows a sample Help auxiliary menu with a cascading menu.

Figure 2-10. **Help auxiliary menu with a cascading menu**



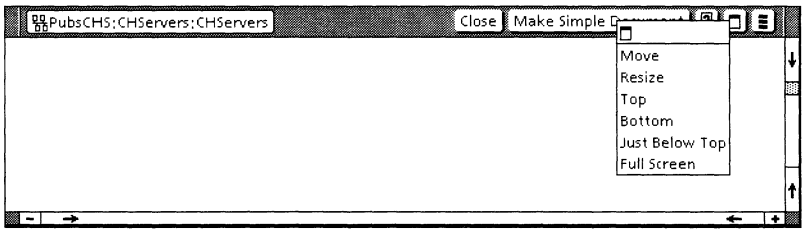
The System Administrator selects a Help auxiliary menu item, and moves the cursor to the arrow next to the item to display a subordinate pop-up menu. When the System Administrator selects an item from the pop-up menu, a Help Text window appears.

Window management commands

Window management commands are consistent with those for other GLOBALVIEW windows. The System Administrator accesses these commands through the Window management auxiliary menu.

Figure 2-11 illustrates the window management commands.

Figure 2-11. **Window management commands**



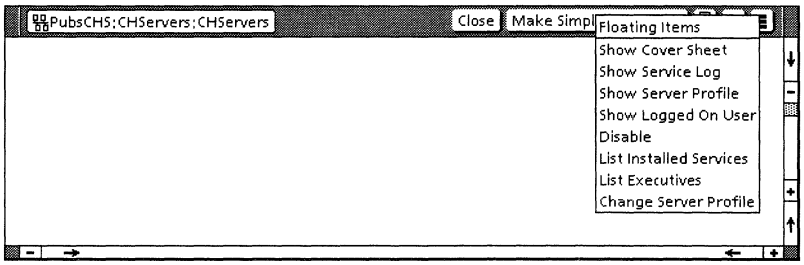
For a complete discussion of window management commands, refer to the *GLOBALVIEW General User Guide*.

Server commands

The Service Executive provides a set of ten commands known as server commands. These commands are contained in the Floating Items auxiliary menu of the Service Executive window, and are available to all services.

Figure 2-12 illustrates the server commands.

Figure 2-12. **Server commands**



The following is a list of the commands and a brief description of what each command does:

Show Cover Sheet Allows you to display a cover sheet. This command is not necessary to manage servers or services. You may use the cover sheet to record notes about the server and the services installed on it.

Don't Show Cover Sheet Allows you to close an open cover sheet. This command is not necessary to manage servers or services.

Show Service Log Displays a Text window showing the contents of the service log. The service log contains information about recent service activity.

Show Server Profile Displays the server profile. The server profile contains the following items of information about the server: Network Number, Server Name, Server Description, Remote Execs, Workstation Timeout, and Log File Size.

Show Logged On User Displays the name of the System Administrator and indicates whether logon status is enabled or disabled.

Enable Allows the System Administrator to use special server and service commands (such as changing the server profile or stopping the service).

Disable Makes special server and service commands unavailable.

List Installed Services Lists the services running on the server.

List Executives Lists any other System Administrators who are conducting maintenance sessions with the server.

Change Server Profile Allows the System Administrator to alter five fields in the server profile: Server Name, Server Description, Remote Execs, Workstation Timeout, and Log File Size.

Software components

The Service Environment is comprised of two software components: SEExecutive and SECore.

- SEExecutive provides the Graphical User Interface (GUI) for managing services. It also allows the System Administrator to access a server and services from a remote location, usually a workstation.

  SEExecutive must be installed on any workstation from which the System Administrator intends to manage services.

- SECore manages communications between a service and network clients (workstations and other services). The interfaces to clients are modules of code known as NSExec, TextIO, and SelfReg.

  — NSExec allows clients to self-register as clients of the Service Executive.

  — TextIO is called by clients to obtain user input for commands, and to display messages to the user.

  — SelfReg allows clients to obtain a service name from the user, and to register that name with the Clearinghouse Service.

  SECore should only be installed on workstations which are used as servers.

Command structure

The Service Environment defines a method of command processing followed by all services. Server and service commands are composed of phrases, predicates, and call-back procedures (also referred to as programs).

The individual services provide phrases, predicates, and call-back procedures for their operations. The Service Environment provides phrases, predicates, and call-back procedures for server commands.

These command elements are defined as follows:

- A phrase is two or more key words which label a command, such as [Add User] or [Delete Mailboxes].
- A predicate determines whether a command should be made available to the requesting client.
- A call-back procedure is the client software that is executed when the command is invoked.

The Service Environment evaluates the predicate for client commands, and then makes a phrase available to the requestor. When the requestor selects a phrase, the Service Environment then executes the appropriate client call-back procedures.

Phrases

Command phrases are composed of two or more key words. The first word is usually a verb, and the remaining words are the object of the verb. Typical verbs used as commands are [Add], [Change], [Delete], [List], and [Show]. One exception is the non-verb [Service], which the Service Environment treats as a verb.

The Service Environment has a standard verb set.

Service Groups all commands that deal with a service state, such as Start, Stop, Expunge, Backup, and Restore.

Add Adds an object to a service database.

Change Modifies a database object.

Delete Removes a database object.

List Displays a list of one or more database objects.

Show Displays information about an object in ReadOnly format.

Client-defined verbs are used only when needed. For example, the Print Service requires a command to print a test pattern. The Print Service therefore has a [Print] button with the object [Test Pattern] in the button's pop-up menu. This is referred to as the [Print Test Pattern] command.

Access to commands

For server commands, the entire command phrase appears in the Floating Items auxiliary menu of the Service Executive window (for example, [Show Log] and [List Executives]).

For service commands, the first word of a command phrase is displayed on a button in the Service command subwindow of the Service Executive window.

The System Administrator points to a command button and holds down the left mouse button. A pop-up menu displaying related objects appears. This technique conserves screen space and groups commands logically.

Predicates

The predicate allows the requestor to interact efficiently with the executive. A predicate determines whether the command is available to the requestor.

The predicate is frequently a function of whether or not the System Administrator is enabled, but it can also be a function of other variables such as whether the service is started or stopped. When a command is executed, the predicate is evaluated.

Call-back procedures

When the System Administrator selects a command and the command is available, the Service Environment executes a call-back procedure supplied by the client. The typical uses for call-back procedures are to provide status information, update databases, and control software.

Normal startup

After the System Administrator creates a Clearinghouse Service (in Genesis mode) and starts the server for the first time, virtually all subsequent startups are normal.

Server

SECore runs and successfully self-registers with the Clearinghouse Service. It obtains the server name and description from the server profile.

If the server's name and description are already in the Clearinghouse they are verified.

Service

Each service self-registers with the Clearinghouse Service by extracting the service's name and description from the service profile.

If the service's name and description are already in the Clearinghouse (which is expected), then they are simply checked.

Typically, each service starts and makes itself available to the network without any interaction by the System Administrator.

If the service's StartAfterRunning boolean is set to TRUE (which is the usual case), then the service executes its own start procedure. If the StartAfterRunning boolean is set to FALSE, the service remains stopped.

Some services allow the System Administrator to request a non-normal startup, using non-normal startup switches. If the System Administrator has done so, the service remains stopped.

TTY interface

The TTY interface is the vehicle for command entry for the File Service and a number of general server commands. The interface is supplied by software called Services Common Software for UNIX (SCSX).

This section presents the following subjects:

- Purpose
- User interface
- Command entry shortcuts
- Command parsing
- Command structure.

Benefits

The TTY interface and underlying SCSX software, which are required to run the File Service, have several benifits:

- The TTY interface fills the need for an easy-to-use method of command entry; it is the same as the 8000 and 8090 servers and the MS-DOS and UNIX method of command entry.
- The System Administrator can use the UNIX **rlogin** command to access SCSX and File Service software from a different workstation and manage the File Service remotely because the software runs as a UNIX application. The System Administrator can also manage the File Service from its physical location.
- If more than one System Administrator logs in to manage a File Service, each executive session is a separate UNIX activity.

User interface

This section describes TTY usage concepts. For a step-by-step orientation to TTY interface operations, refer to the File Service chapter of the *System Administration Reference*.

Entry and exit

The operator enters the TTY interface by suspending GLOBALVIEW. At the UNIX prompt, the operator logs in to UNIX account **xnsadm**. This executes a captive login procedure; the login script automatically executes the SCSX software. The operator is now in the TTY environment.

The operator leaves the TTY environment by using the SCSX command, **Quit Connection**. At the UNIX prompt, the operator enters **xgvuser** to return to GLOBALVIEW.

Command line prompts and user status

The TTY interface displays different command line prompts, depending on the operator's user status. User status varies, depending on the operator log on and enable status.

Ordinary user status

The operator has entered the TTY environment, but has not logged on. The system displays the following prompt:

```
>
```

Logged–on user status

The operator has issued the **Logon** command, but is not enabled. The system displays the following prompt:

```
>
```

Enabled user status

The operator is logged on, and has issued the **Enable** command. The system displays the following prompt:

```
!
```

Command parsing and evaluation determine the validity of an entered command (for example, whether **enable** is spelled correctly). Authentication determines the operator's System Administrator privileges (for example, whether the user is a System Administrator). The TTY interface software then provides the appropriate command line prompt.

Different commands are available, depending on user status. For example, enabled user status provides access to the commands with the greatest impact on data and service performance.

Command line prompts and service context

A context is a logical grouping of commands by service. To use a service's commands, the operator must be in the service's context. When the operator is in the context of a service, only that service's commands and SCSX commands may be executed.

The operator may enter a service context, change a service context, or reset the service context to no context (also known as the server context). For further information on performing these procedures, refer to the *GLOBALVIEW Document Services System Administration Reference*.

The context mechanism is similar to the Service Executive windows in the Service Environment, which differentiate between the types of commands that you can perform, such as server commands, window commands, and service commands.

Each context has an abbreviation, which is displayed in the command line prompt. The abbreviation corresponds to the service being managed (for example, FS for File Service).

The operator issues the **File Service** command to enter the File Service (FS> or FS!) context. The operator issues the **Reset Context** command to return to the server (> or !) context. The following are examples showing different command line prompts in the File Service context, depending upon user status.

Ordinary user

When the operator is not logged on, but has selected the File Service context, the system displays the following prompt:

```
FS>
```

Logged-on user

When the operator is logged on, is not enabled, but has selected the File Service context, the system displays the following prompt:

```
FS>
```

Enabled user

When the operator is logged on, is enabled, and has selected the File Service context, the system displays the following prompt:

```
FS!
```

This prompt specifies that the system is ready to accept appropriate commands. The following example shows the entry of a File Service command.

```
FS!Add File Drawer
```

Command parsing and evaluation determine the validity of entered commands (for example, whether it is a File Service or SCSX command).

TTY prompts

The TTY interface provides the operator with several types of prompts to elicit a variety of responses. The operator responds to TTY interface prompts by typing the information requested and pressing the <Return> key.

The TTY interface prompt types include the following:

- Yes/No
- Choice
- Text
- Name
- Decimal.

The following shows a Yes/No prompt. The only acceptable entries are **Y** (Yes) or **N** (No).

```
Continue (Y/N)?: N
```

In the above example, the system has supplied a default value. The operator may accept the default value by pressing <Return> or may type **Y** and press <Return>.

The following example shows a choice prompt:

```
Select Volume
1   Finance
2   Planning
Enter choice number:
```

The operator may select one of a number of choices.

The following example shows a text prompt:

```
File drawer name: Accounting Forms
```

The following example shows a name prompt:

```
User name: Jim J. Jones:Unit 3:Acme
```

Authentication determines the validity of the operator's name.

The following example shows a decimal prompt:

```
Page Limit (type 0 for no limit) (0..2147483647): 10000
```

In many entry situations (especially **List** commands) the operator may use the wildcard character (*):

```
File drawer name: *
```

The software will evaluate the entry and operate on all objects whose names match the entered pattern (for example, all file drawers).

In all cases, the software is capable of evaluating the operator responses for type, length, range, and authenticity. When user input is unacceptable, the system provides error messages and an opportunity to enter the data again.

Display continuation

When the operator issues a command to display information, and there are more than 20 lines of output, the system pauses after displaying 20 lines and displays the following prompt:

```
(More?)
```

The operator responds by pressing the space bar. The next 20 lines of information display. If there are still more than 20 lines to display, the system displays the prompt again. This continues until the display is complete.

Command entry shortcuts

There are several shortcuts that facilitate entry of TTY commands.

Entering part of a command

The operator does not need to enter the command. It is necessary only to enter part of the command and press the space bar. The system recognizes the characters entered and displays the entire word. The operator must enter enough of the command to make it unique.

This example shows entering only part of the second word in a command:

```
FS!List  vo<space  bar>
```

The system displays the complete command as follows:

```
FS!List volumes
```

The operator can then press <Return> to execute the command.

The operator can also enter part of the command word and just press <Return>. If the operator does this, the command will expand and execute, as shown below:

```
FS!List    vo<Return>
```

Erasing a command

To erase part of a command, the operator presses the <Backspace> key. Each time the operator presses <Backspace>, the system deletes one character. Holding down the <Control> key and pressing the <H> key produces the same result.

Canceling a command

To cancel a command, the operator holds down the <Control> key and presses the <C> key. The system cancels the command and does not not execute it, as shown below:

```
FS!List volumes<Control><C>
Command canceled.
FS!
```

Listing available commands

The operator can display a list of available commands by entering a question mark (?). The screen displays the available commands, as shown below:

```
FS!?
Add, Change, Create, Delete, List Offline, Online, Reset,
Show
FS!
```

If only part of a command is known, the operator can enter a question mark and the portion of the command to display a list of commands that begin with the part of the command entered. In the following example, the operator knows that objects can be listed, but wants to find out the object types:

```
FS!List ?
List Desktops, List File Drawers, List Volumes
FS!
```

The file types returned are desktops, files drawers and volumes.

Command parsing

The command parsing mechanism for the TTY interface makes possible most of the command entry shortcuts. In addition, the interface software validates command entry, ensuring that only available commands are selected.

The software captures each character as it is entered. Characters are captured until the interface software detects that <Return> has been pressed or another event occurs. Input analysis and validation typically operate according to the following:

- If the entry is the <Control><C> combination, the command is canceled. The system displays the message"Command canceled", advances two line feeds, and redisplays the command line prompt.
- If the entry is the <Control><H> combination or <Backspace>, the interface software deletes the last displayed character from the screen and the input buffer, and positions the screen cursor next to the last displayed character. If <Control><H> or <Backspace> is the first entry on the command line, no action occurs.
- If the entry is a question mark (?), the interface software performs the following additional functions:
  - — If the question mark is the first character entered on the command line, the interface software displays the complete list of available commands.
  - — If the question mark is not the first character on the command line, the interface software displays the list of commands matching the pattern entered (for example, **St?** produces a list showing **Start Services** and **Stop Services**).

  - If the question mark is not the first character on the command line, and a command verb has been displayed, the interface software displays the list of objects the verb may act upon (for example **List ?** produces a list showing **List Desktops**, **List File Drawers**, and **List Volumes**).

- If the entry is a hard space (press the space bar), the interface software performs the following additional functions:
  - If the space is the first character on the command line, no command expansion is performed.
  - If the space is not the first character on the command line, the interface software tests the entered characters against the list of valid commands and performs the following.
    - If there is a match, the interface software expands the complete name of the command on the screen.
    - If there is no match, or the match is ambiguous, the interface software displays a question mark on the screen.
- If the entry is the <Return> key, the interface software performs the following additional functions:
  - If the <Return> key is the first character on the command line, no action occurs.
  - If the <Return> key is not the first character on the command line, the interface software tests the entered characters against the list of valid commands and performs the following:.
    - If there is a match, the interface software displays the complete name of the command on the screen, and executes the command.
    - If there is no match, or the match is ambiguous, the interface software displays a question mark on the screen.

This validation process filters cancellations, requests for help, and erroneous entries. The result is valid command entries that the system can execute.

Command structure

A command has three components: a predicate, a phrase, and a program (also known as a call-back procedure).

- A predicate determines whether a command should be made available to the requesting user.
- A phrase is a series of words called keywords, which the user enters to invoke a command.
- A program is the software which is executed when a command is invoked.

The SCSX software evaluates the predicates, matches user input to an inventory of phrases, and calls the appropriate program. Predicates, phrases, and programs,are provided by the individual services.

Predicates

The predicate allows the operator to interact efficiently with the interface. A predicate determines whether a command is available to the operator.

The predicate is frequently a function of whether or not the System Administrator is enabled, but it can also be a function of other variables such as whether the service is started or stopped. When the System Administrator executes a command, SCSX evaluates the predicate. If the System Administrator's privilege level (enabled or disabled), or the service state (started or stopped) changes, the interface can display a new set of commands to reflect those changes.

Phrases

Command phrases are composed of two or more words, known as keywords. The first word is usually a verb, and the remaining words are the object on which the verb operates. Typical verbs used as commands are **Add**, **Change**, **Delete**, **List**, and **Show**.

One exception to this rule occurs when the object of the verb is clearly implicit in the verb itself (for example, **Logon**). Another exception is the selection of service context (for example, **File Service**).

Programs

When the System Administrator selects a command, the SCSX software calls a program supplied by a service.

The typical uses for programs are providing status information, updating databases, and controlling the software.

User interaction with a program may be very extensive or practically nonexistent. It usually consists of a sequenced input of parameters followed by internal processing (for example, deleting a desktop) and the output of characters to the screen (for example, listing an object or displaying the message, "Done").

The processing may range from very minimal to very extensive. Processing may occur between entering parameters, or after all parameters are entered.

Command summary

This section contains a brief summary of the commands which are available through the Service Executive interface, TTY interface, and UNIX.

Service Executive server commands

Show Cover Sheet	Displays a cover sheet.
Don't Show Cover Sheet	Reverses the effect of the [Show Cover Sheet] command.
Show Service Log	Displays a Text window showing the contents of the service log.
Show Server Profile	Displays the server profile.
Show Logged On User	Displays the name and status (enabled or disabled) of the logged on System Administrator.
Enable	Allows the System Administrator to use privileged server and service commands.
Disable	Reverses the effect of the [Enable] command, making privileged server and service commands unavailable.
List Installed Services	Lists the services installed on the server.
List Executives	Lists other System Administrators who are conducting maintenance sessions with the server.
Change Server Profile	Allows the System Administrator to alter the server profile.

Service Executive buttons

Service	Accesses service-related commands. This is a standard button.
Add	Adds a database object. This is a standard button.
Change	Changes a database object. This is a standard button.
Delete	Deletes a database object. This is a standard button.
List	Displays a list of similar database objects. This is a standard button.
Show	Displays details about a database object. This is a standard button.
Reset	Resets statistics. This is a Print Service button.
Print	Prints test patterns. This is a Print Service button.
Cancel	Cancels jobs. This is a Print Service button.
Release	Releases jobs. This is a Print Service button.

SCSX commands

Enable	Allows the System Administrator to use privileged server and service commands.
Disable	Reverses the effect of the [Enable] command, making privileged server and service commands unavailable.
Expunge Service	Removes the File Service from the server.
Install Service	Installs the File Service on the server.

List Services	Lists the services installed on the server.
Logoff	Changes the status of the operator from logged on user to ordinary user.
Logon	Changes the status of the operator from ordinary user to logged on user.
Quit Connection	Ends the session and returns the operator to the UNIX environment.
Register Server	Adds a server to the Clearinghouse Service database.
Reset context	Exits a service context and returns the operator to the server context.
Show Logged On User	Displays the name and user status (enabled or disabled) of the logged on user, normally a System Admintrator.
Show Server	Displays information in the server profile.
Show Time	Displays the current date and time.
Start Services	Starts the File Service after a **Stop Services** command.
Stop Services	Stops the File Service for special File Service operations.
Unregister Server	Deletes a server from the Clearinghouse Service database.

TTY entry techniques

<Backspace>	Backspaces and deletes one character.
<Control><C>	Cancels a command.
<Control><H>	Backspaces and deletes one character.
<Space bar>	Expands a partial command entry.
<Space bar>	Causes another screen of data to display on the terminal.
?	Displays available commands.
*	Allows the operator to manipulate multiple objects with similar names. This is a wildcard character.

UNIX commands

login	Allows the operator to log in to the UNIX account xnsadm to perform TTY-oriented File Service operations.
logout	Allows the operator to log out and returns the operator to the UNIX login prompt.
rlogin	Allows the operator to log in to a remote workstation or server using the UNIX account xnsadm to perform TTY-oriented File Service operations.
xnsadm	Allows the user to manage the File Service. This is a login account.
xgvuser	Returns the operator to GLOBALVIEW. This is a login account.

Terminology

boot To start or restart a server or workstation using a UNIX command. Also the act of causing the server to clear its main memory, and reload the operating system and the server software.

command Top-level mechanism for causing a system administration operation to take place.

context Command grouping mechanism. In the TTY interface, the System Administrator selects which service to manage by entering the context of that service.

executive session General term for the System Administrator's link to network services. For the File Service, an executive session has a TTY user interface; other services have a Graphical User Interface.

Graphical User Interface Method of command selection using windows, pop-up menus, and options sheets. The Service Executive has a Graphical User Interface.

parsing Function of the Service Environment or SCSX which analyzes command input to determine its nature and validity.

register To create or check a Clearinghouse Service entry for a server or service. This operation typically takes place at server or service startup.

restart Restarting the runtime environment, the Basic Workstation Software, and GLOBALVIEW applications which execute automatically upon startup. The user uses a UNIX command to restart the environment -- typically **xgvuser**, which activates the GLOBALVIEW environment, the Service Environment and services. Restart does not refer to a UNIX operating system reboot.

SA *See* System Administrator.

SCSX Services Common Software for UNIX. Provides the TTY interface and underlying software for the File Service.

SE *See* Service Environment.

SECore Application on the server. SECore implements the NSExec, TextIO, SelfReg, and private interfaces. Clients (workstations and other services) use SECore to communicate with users and with the Clearinghouse Service. The presence of SECore defines a workstation as a workstation or server.

SEExecutive BWS application that supplies the Graphical User Interface for network services. SEExecutive can reside on the System Administrator's workstation, on a server, or on both. The SEExecutive application communicates with the SECore application using a special protocol.

server profile Data element known as an optionfile containing server information such as server name, server description, and network number. Service Environment and SCSX commands are available to change some of the profile entries.

Service Environment General term for the underlying software that permits several network services (Clearinghouse Service, Mail Service, and Print Service) to run on a Sun-based workstation or server. The Service Environment is the successor to Services Common Software (SCS), used on Xerox 8000 and 8090 servers.

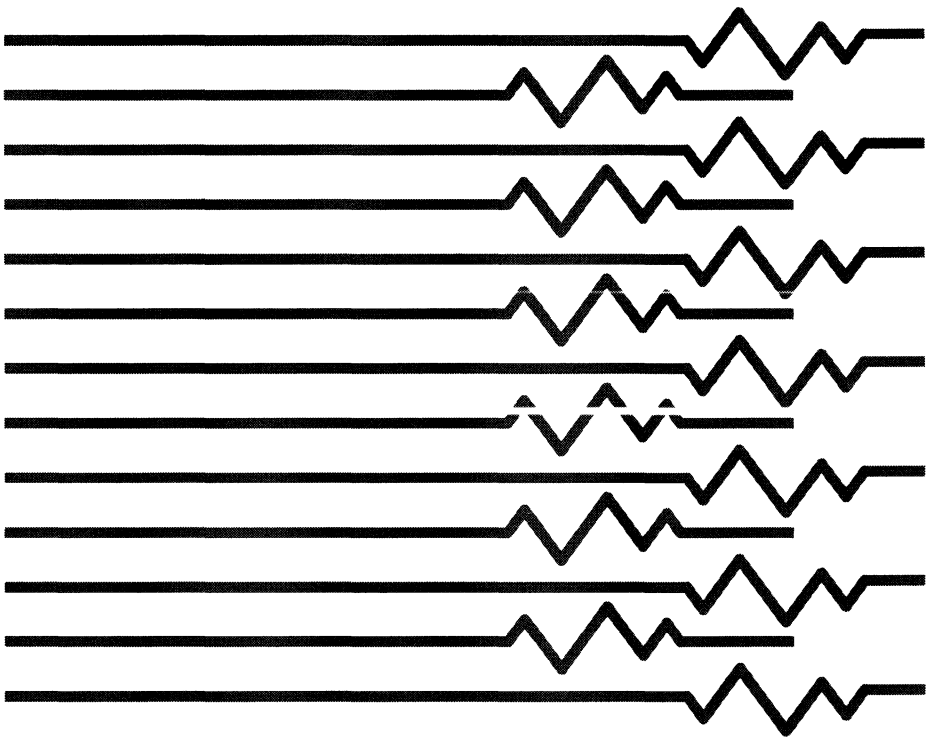
service profile Data element known as an option file containing service information such as service name, service description, and configuration data. A service typically provides its own command to change the service profile entry. Each service has one service profile.

System Administrator Individual responsible for managing servers and services in a domain.

TTY Abbreviation for teletypewriter.

TTY interface Command line-oriented interface used by SCSX and the File Service.

Clearinghouse Service



3. Clearinghouse Service overview

Introduction

A modern information handling system must be able to store and catalog all of the objects that exist within that system, to quickly and efficiently locate specific objects, and to modify or act upon those objects.

The Xerox Clearinghouse Service manages the lists of people, groups, and services that comprise a Xerox Network Services (XNS) network.

Overview of the Clearinghouse Service

The Clearinghouse Service is a highly specialized database that allows network clients to easily locate resources and registered objects within an XNS network. The Clearinghouse Service stores and keeps track of information about the users and services on a network. The stored information is collectively called the Clearinghouse database. The Clearinghouse database is an assembly of names, network addresses, and resource attributes.

Each Clearinghouse database contains both local and global database information. The local portion of a Clearinghouse Service database contains information specific to the local domain, such as users and services that reside in the domain. The global information stored in a local Clearinghouse Service database contains a list of the domains and Clearinghouse Services that serve those domains, and the addresses of all the Clearinghouse Services in the network

The Clearinghouse Service provides a directory service, much like the telephone book, that answers client (workstation software) questions regarding network resources. Through the directory service, clients at one end of a network can quickly locate resources at the other end of a network. The complex task of resource location is transparent to the user.

The Clearinghouse Service also functions as an authentication service. By cross–referencing names and passwords stored in the database, the Clearinghouse Service checks and validates user identification. Without Clearinghouse Service validation, clients and users are locked out of private resources such as file drawers and mail boxes.

Routine database maintenance tasks include such things as adding new user names, removing obsolete names, and changing passwords. The Clearinghouse Service has a set of commands that allow System Administrators to add, modify, and delete all

objects registered in a Clearinghouse Service database except other Clearinghouse Services.

Database replication

The ability of the Clearinghouse Service to replicate its database ensures that the System Administrator will be able to have at least two copies of each domain existing concurrently on-line. If a Clearinghouse database is damaged or destroyed, the copies are unaffected and still able to serve client and user needs.

Database replication is the basis for a distributed system. The database for Domain 1 may exist not only on the local Domain 1 Clearinghouse Service, but can also be replicated on the local Domain 2 and Domain 5 Clearinghouse Services. If the database on Clearinghouse Service 1 is destroyed, copies of Domain 1 are available from other Clearinghouse Services.

If the local Clearinghouse Service for Domain 4 is unavailable, the client uses the network information received from the local database to determine where the database for Domain 4 is replicated. It then connects to that Clearinghouse Service to obtain the information it needs.

Replication also enhances the response time normally required to access a domain database. The more objects and users contained within a domain, the more often it will be accessed. If a single Clearinghouse Service serves a very large domain, that service may become so busy processing requests that its response time may markedly deteriorate. If a domain database is replicated in several locations, the burden is distributed, thereby decreasing access time to a single database and increasing efficiency.

Clearinghouse components and service design

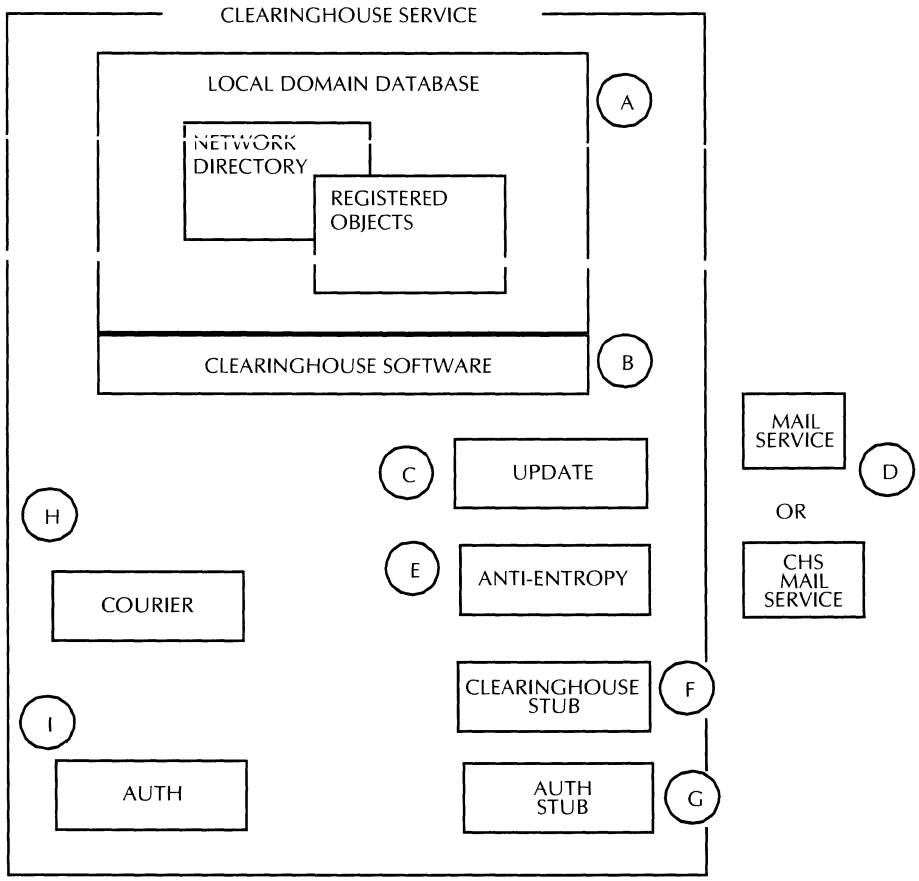
This section presents an overview of the design of the Clearinghouse Service, the modules that make up the Clearinghouse Service, and how the modules fit together.

Modules are the individual elements of the overall structure of a Clearinghouse Service. Modules may be part of the service software package, service protocols, or essential peripheral elements.

Software modules

The Clearinghouse Service design, shown in Figure 3–1, is a functional schematic of the service and is not intended to show actual positions or relative size of software modules or protocols. For the sake of clarity, service software, protocols, and peripheral elements are treated as individual software modules.

Figure 3-1. **Clearinghouse components**



The Clearinghouse Service uses the following modules:

A. Clearinghouse database
B. Clearinghouse software
C Update

D. Degenerate Mail Service
E. Anti-entropy
F. Clearinghouse stub
G. Authentication stub
H. Authentication
I. Courier

Clearinghouse database

The Clearinghouse database stores individual data elements called objects. The entire collection of objects form the Clearinghouse database.

The Clearinghouse software manages the database. Functions of the service software include database query and object creation, modification, and deletion.

Each Clearinghouse has responsibility for part of the entire network database. The part maintained by each Clearinghouse Service is called the local database. A local database can support any number of organizations, domains, and objects.

While organizations may be distributed among several Clearinghouses, domains must be contained within a local database. If any part of a domain is maintained by a local database, then all of it must be maintained by that local database. Domains may be replicated on other Clearinghouse Service databases.

A local Clearinghouse Service is referred to as a Domain Clearinghouse. A Domain Clearinghouse contains data mappings for its own database and a map to the other domains in the organization.

The Clearinghouse is not a general purpose shared database system. It is organized to efficiently perform its role as a resource locator. The database is designed for high performance information query and low performance object modification.

Search order

Objects in the database may be created and deleted while the Clearinghouse is responding to database queries. The client must assume that database information may change between operations.

When a pattern is used in a database operation, the operation stops on the first object it finds that matches the pattern.

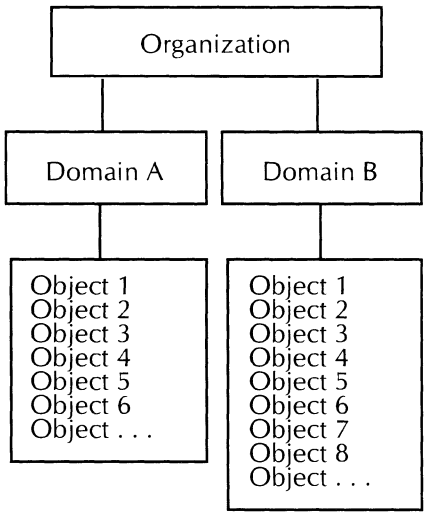
Database objects

A database object consists of a name and associated property list. An object entry in the database is structured as a set of triples: object name $\rightarrow ID_1, type_1, value_1$.

This set of triples is the properties of an object. Each triple consists of a property ID number, a property type, and a property value.

Object naming—As shown in Figure 3–2, the organization of the Clearinghouse database is based on a hierarchical naming structure. The organization name is at the top of the hierarchy. Under the organization are a number of domains, and under each domain are numerous registered objects.

Figure 3-2. **CHS hierarchical naming structure**



Three–part name—Every network object must have a unique network name. The name of an object reflects the hierarchical structure of the network. This is called a "three–part", or "fully qualified", name. A three–part name contains an object name, a domain name, and an organization name; all separated by colons. The format of the three–part name is always Object:Domain:Organization.

Organizations include corporations, universities, and government agencies. Very large corporations may have several organization names. Subsidiaries or foreign affiliates of a larger parent company may have a unique organization name.

The domain name divides the organization. Domain names must be unique to the organization; it is impossible to have two identical domain names within one organization.

The object name is the name of a specific object or user registered in a domain. The object name is also referred to as the local name or the "distinguished" name.

Objects can be either logical or physical entities. Logical objects include file drawers, service volumes, and mailboxes. Physical objects include printers, communications ports, disk drives, processors, and server terminals.

An object name for a user consists of a first name, middle name or initial, and last name. An object name for a service might indicate the service function or geographical location, such as "High Volume Printer" or "Downtown Mail Service."

Object names must be unique to a domain. The Clearinghouse will not register two identical object names within the same domain.

Aliases—The Clearinghouse Service supports the use of aliases. An alias is a nickname for a registered object. The object is still registered under its distinguished name, but you may also refer to it by its alias.

An alias must be unique to a particular domain. The Clearinghouse will not register duplicate aliases in the same domain.

An alias has a single property value that points to another object; the distinguished name and its associated properties. All operations undertaken by the service are performed on the distinguished name, not on the alias.

Patterns—The Clearinghouse can perform basic pattern matching if the wildcard character, the asterisk (*), is used as all or part of the three–part name during the search.

You can replace the object name with an asterisk, replace both the object and domain names with asterisks, or replace all three names with asterisks.

The Clearinghouse software searches the database for objects that match the pattern. Object names are compared on a character–by–character basis from left to right. The search is not case-sensitive. The pattern "A*B" matches any object name starting with "A" and ending with "B;" such as "AbB," "AcqZxweb,", or "AAaab".

Object properties—Each database object has an associated set of properties. Each property has an ID number, a type, and a value. There are two types of properties; an item property and a group property.

The value of an item property may contain any data. The value of a group property is a sequence of names called "members." Clearinghouse Service software is capable of adding, deleting, enumerating, and searching the members in a group property.

Updates

The Clearinghouse Service uses the Clearinghouse Update Protocol to propagate database updates among multiple servers.

Clearinghouse mail service

The Clearinghouse uses the XNS Mail Service to distribute database updates to other Clearinghouse Services. The availability of some form of Mail Service is necessary if the Clearinghouse Service is to send updates.

If a Mail Service is installed on the same server as the Clearinghouse Service, it is used by that Clearinghouse Service to send and receive update messages. If there is no existing co–resident mail service, the Clearinghouse creates a Mail Service to handle the update functions.

The Mail Service software is part of the Distributed Services Common Software application that is installed with the Clearinghouse Service software. Administrators do not need to install or manage a Clearinghouse mail service.

Clearinghouse software

The Clearinghouse protocol is used to perform basic database functions, such as queries and modifications. The Database Access protocol facilitates changes to access rights and inspection of the contents of the database.

Anti–entropy

This is the software that performs the nightly database consistency checks. The consistency check operation involves comparing the daily-updated information, organizations, and domains that Clearinghouse Service databases have in common. Discrepancies are resolved using the data that has the most recent time stamp and a valid checksum.

Authentication

This is the software that performs user authentication.

Authentication stub

This is the front end for the Authentication Protocols.

Clearinghouse stub

This is the front end for the Clearinghouse Protocols.

The Clearinghouse and Authentications interfaces include the software necessary to navigate through the distributed Clearinghouse System and to locate the databases needed to satisfy the interface client.

Courier

This is the protocol the Clearinghouse Service uses to exchange data with network clients.

Functions and operations

This section discusses the five basic functions of the Clearinghouse Service:

- Database modification
- Database queries
- Database updates
- Database consistency checks
- Client authentication.

Database modification

As illustrated in Figure 3–3, System Administrators modify database objects using the GUI executive window. Commands and actions are sent from the GUI and client to the Clearinghouse stub, then to the Clearinghouse software as a Courier remote procedure, and finally to the database.

Results of the modification follow a converse path back to the client and the GUI.

Database queries

The most common Clearinghouse activities are the database enumeration and lookup queries.

An enumeration query allows the network client to obtain names of all entries containing a specified property, such as all users or all File Servers, in a domain.

A lookup query allows a client to retrieve specified properties of a named entry, such as the network address of a File Service or registered workstation.

Remote courier process

Communications with Clearinghouse Services use Courier request–reply protocols. Each request is a remote procedure, as defined by the Courier protocol. Each error condition is considered a remote error.

Large amounts of data use the Bulk Data Transfer Protocol.

Remote procedure calls

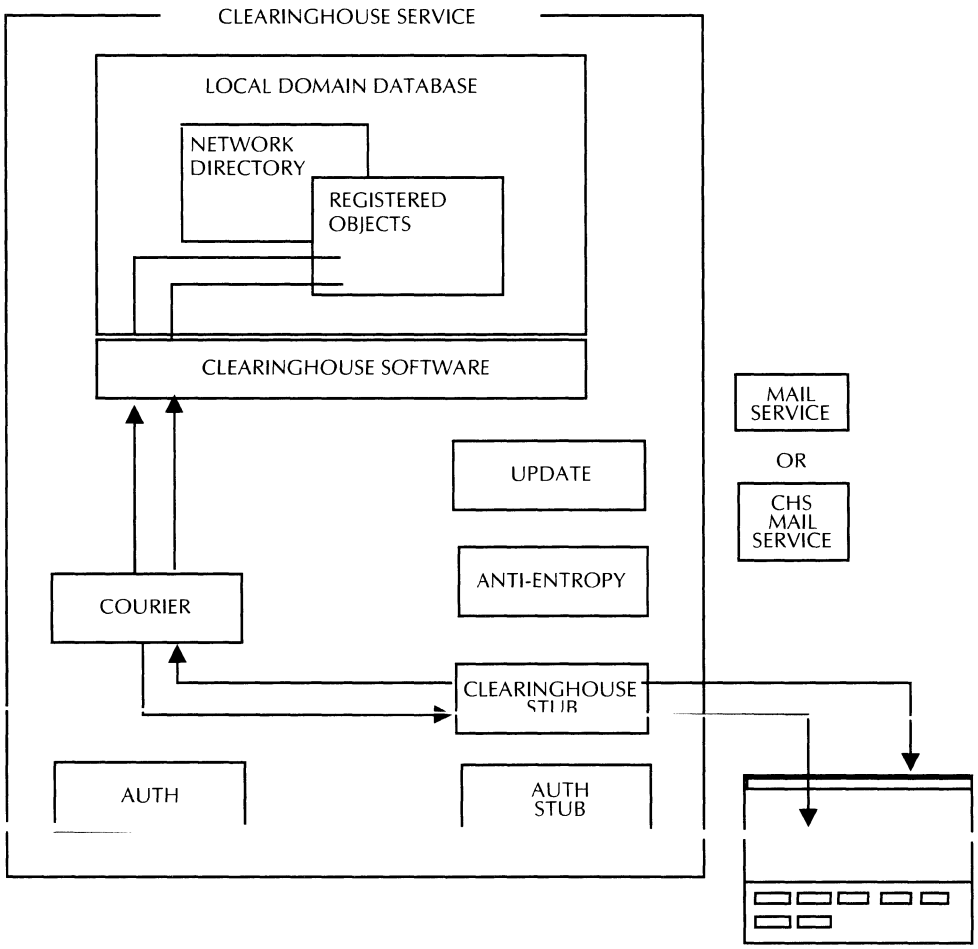
Create object—The CreateObject procedure registers an object in the database. The object has no properties.

Delete object—The DeleteObject procedure removes an object from the database. If the object name is an alias, it is no longer referenced. All aliases that reference an object are also deleted.

Create alias—The CreateAlias procedure attaches an alias to a distinguished name. If the new alias references another alias, that alias is no longer referenced.

Delete alias—The DeleteAlias procedure removes an existing alias from the database, and returns the distinguished name associated with it.

Figure 3-3. **Clearinghouse database modification**



Delete property—The DeleteProperty procedure removes the specified property, both number and value, associated with an object. The property may be of group or type item.

Add item property—The AddItemProperty procedure adds a new item property to an object and gives that property an initial value.

Retrieve item—The RetrieveItem procedure returns the value of the item property associated with an object. The first object name which matches the pattern is used.

Change item—The ChangeItem procedure changes an existing item property.

Add group property—The AddGroupProperty procedure creates a new group property to be associated with an object. The group is initialized with zero or more members as specified at creation.

Retrieve members—The RetrieveMembers procedure retrieves a designated group property value associated with an object.

Add member—The AddMember procedure adds a new member to a group property of an object.

Add self—The AddSelf procedure is the same as the AddMember call, except the name added is that of the user identified by the authentication parameters.

Delete member—The DeleteMember procedure removes a member from a group property of an object.

Delete self—The DeleteSelf procedure is the same as the DeleteMember call, except the name deleted is that of the user identified by the authentication parameters.

Lookup object—The LookupObject procedure returns an object name based on a specified name, partial name, or pattern. When searching the database, the search stops with the first object that matches. If the result of the search is an alias, the referenced object is returned.

List organizations—The ListOrganizations procedure enumerates all organizations available to the Clearinghouse database.

List domains—The ListDomains procedure enumerates all domains in a specified organization.

List objects—The ListObjects procedure enumerates all objects in a domain that match a specified property.

List aliases of—The ListAliasesOf procedure enumerates all aliases associated with an object.

List aliases—The ListAliases procedure enumerates the aliases of all objects in a domain.

List properties—The ListProperties procedure returns the property numbers and distinguished name of a specific object.

Is member—The IsMember procedure determines if an object name is a member of a group property. This procedure has the following operation modes:

- Normal mode -- compares a specific name with each entry in a group property of an object.
- Group mode -- extends the search of an object name to the group properties of objects whose names were in the initial group property.

Making a query

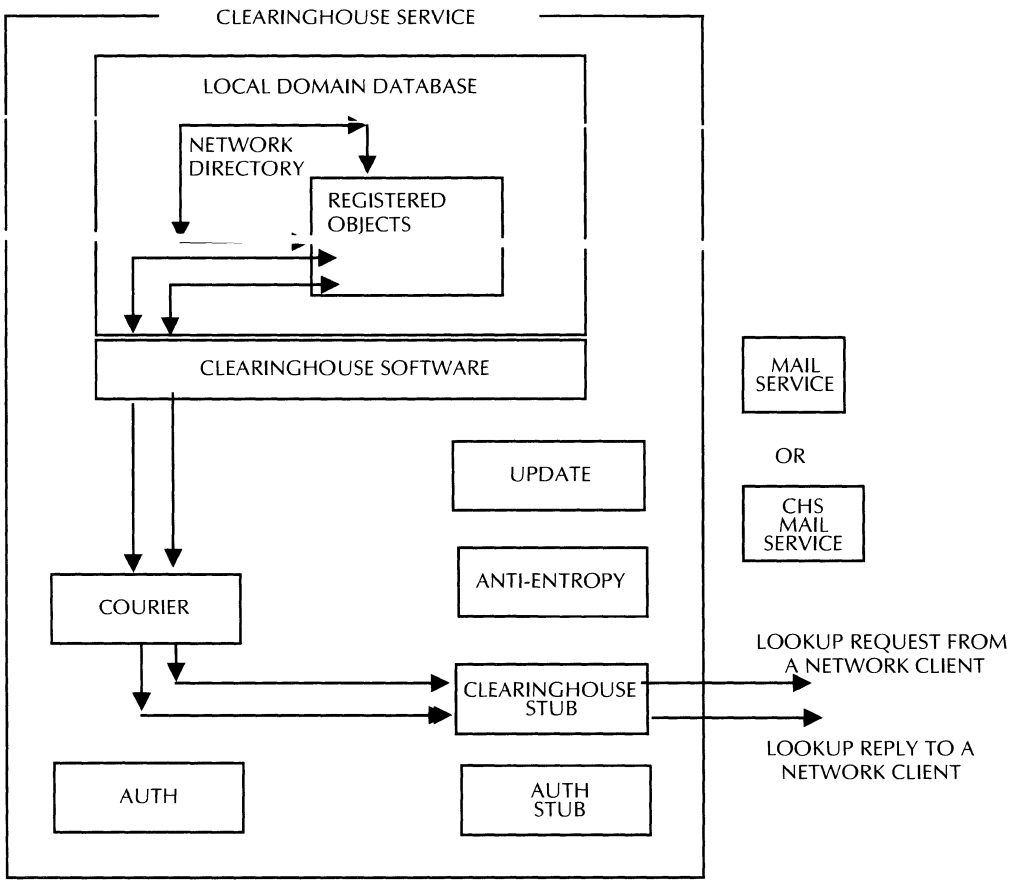
When making a query, a client must locate the network address of the Clearinghouse Service containing the required data. Using an expanding ring broadcast, the client establishes initial contact with a Clearinghouse stub in the distributed system. Once connected, the client requests the data. If the Clearinghouse that the client is connected to does not contain the data, the Clearinghouse will return redirection information.

Information is structured so that in a maximum of three attempts, the client will locate a Clearinghouse Service which contains the desired data. Generally, the client attempts to access the nearest running Clearinghouse Service that can satisfy the request.

As illustrated in Figure 3–4, the client connects to the Clearinghouse stub, and makes the request. The Clearinghouse stub sends the request to the Clearinghouse software as a Courier remote procedure, and finally to the database where the query is initiated.

Results of the query follow a converse path back to the client.

Figure 3-4. **Clearinghouse database client query**



Caching

To eliminate repetitive processes, each Clearinghouse caches network addresses of frequently used Clearinghouse servers. The cache is not maintained or updated by the Clearinghouse. Address changes are updated by the Clearinghouse stub after the Clearinghouse sends out an invalid address from its cache. The operation of the cache is transparent to the client software and users.

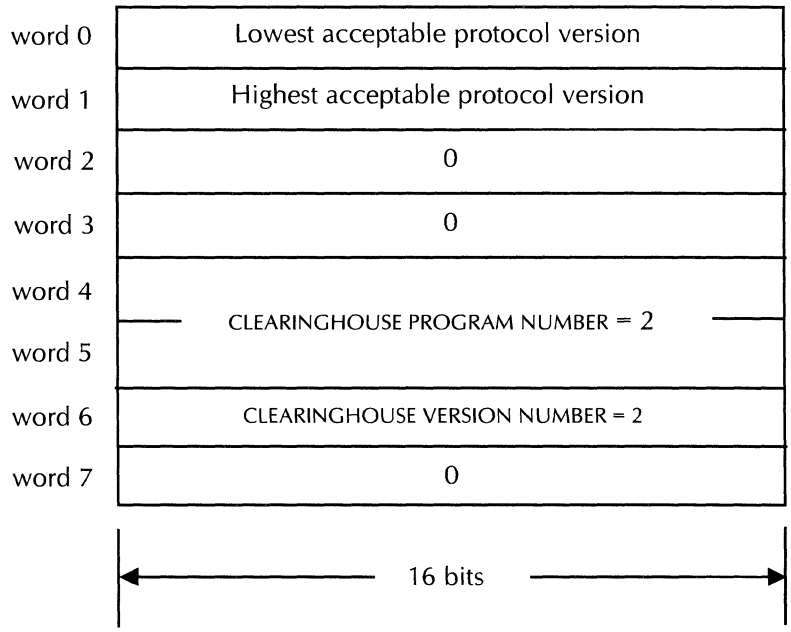
Packet exchange formats

Although information exchange between the client and Clearinghouse Service generally uses the Courier Protocol, the Packet Exchange Protocol is routinely used for certain operations.

BroadcastForServers—The client executes a BroadcastForServers call wheneve it is trying to locate any Clearinghouse Service. Each available Clearinghouse Service on the network responds with its network address.

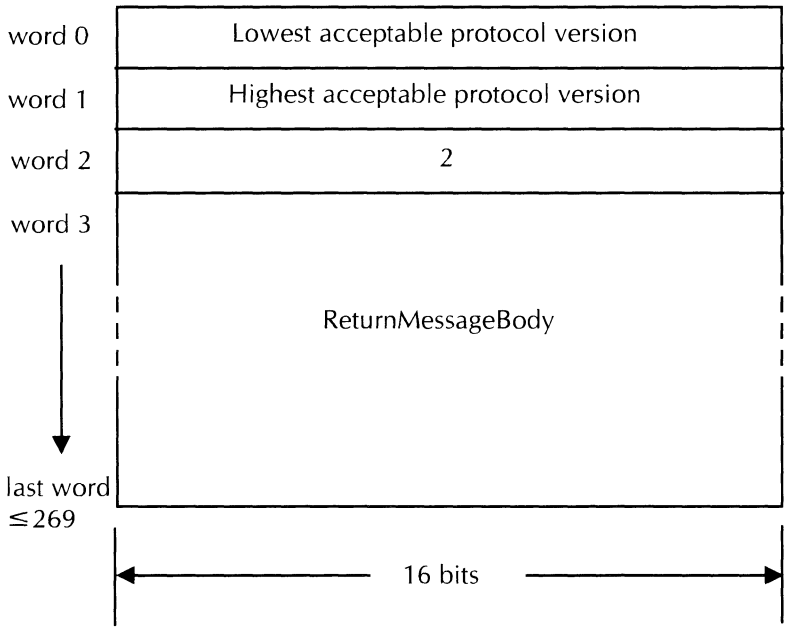
The BroadcastForServers request is a standard Packet Exchange operation. As shown in Figure 3–5, the lowest and highest acceptable protocol versions indicate how the results are encoded. The values of these fields correspond to the Courier version. The answering Clearinghouse Services encode their response according to the Courier version in the highest acceptable protocol version field.

Figure 3-5. **BroadcastForServers request packet format**



The response packets generated by a BroadcastForServers request are shown in Figure 3–6. The lowest and highest acceptable protocol versions indicate how the response is encoded. The remainder of the packet, the ReturnMessageBody, contains the address of the responding Clearinghouse Service.

Figure 3-6. **BroadcastforServers response packet format**



Expedited procedure calls—Some remote procedure calls can use the Packet Exchange Protocol. These procedures are called expedited procedure calls. Clearinghouse operations can use expediting if the following conditions are met:

- The arguments and results must fit into a single Packet Exchange packet.
- The operation must not have a bulk transfer argument.

Because of the size restriction, not all Clearinghouse procedures can use expedited calls. Expedited procedures produce the same results and report the same errors as a Courier operation.

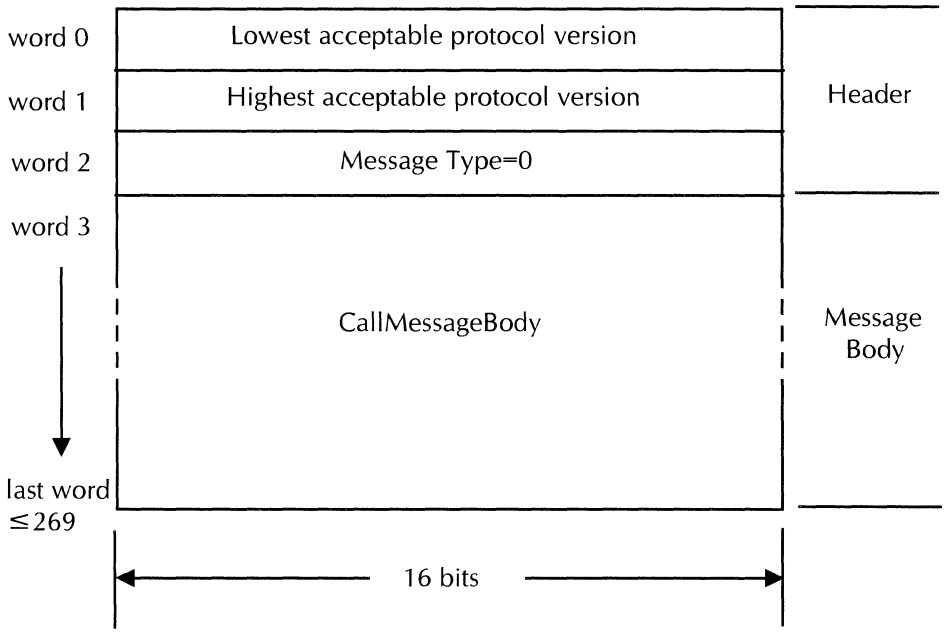
Request packet—As illustrated in Figure 3–7, the expedited request packet has a header and a message body.

The first two fields (words 0 and 1) contain the highest and lowest Courier version acceptable to the client.

The third field (word 2) is the Courier message type. The value is 0 for a request packet.

The rest of the packet is the message body.

Figure 3-7. **Expedited request packet format**



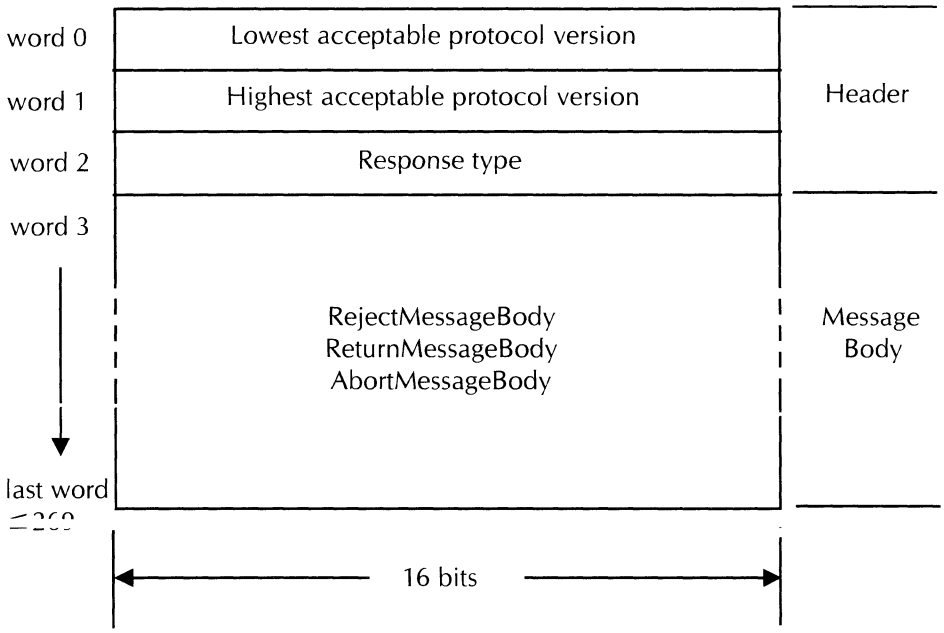
Response packet—As shown in Figure 3–8, the response packet has a header and a message body.

The first two fields (words 0 and 1) contain the highest and lowest Courier version acceptable to the responding server.

Field three (word 2) is the response type. Response type 1 means the server has rejected the request. Response type 2 means the call was successful. Response type 3 means the server is reporting an error.

The rest of the packet is the message body.

Figure 3-8. **Expedited response packet format**



Database updates

Changes that users make to a local domain database are propagated to other Clearinghouse databases where the domain is replicated. The Clearinghouse software generates database updates as simple mail notes, and sends them out through either the network mail service or the Clearinghouse Service mail service.

Changes to the various databases do not occur simultaneously. Race conditions can exist among multiple domain servers. Database inconsistencies result from delayed receipt of update mail messages.

Database inconsistencies are generally temporary. If a domain Clearinghouse Service receives two contradictory updates, it uses the message with the latest timestamp, and adjusts its local database according to the information in that message.

If a Clearinghouse Service receives an update inconsistent with its own database, the inconsistency may be due to lost or delayed update messages, and the new information is added to the Clearinghouse Service. For instance, if a Clearinghouse Service

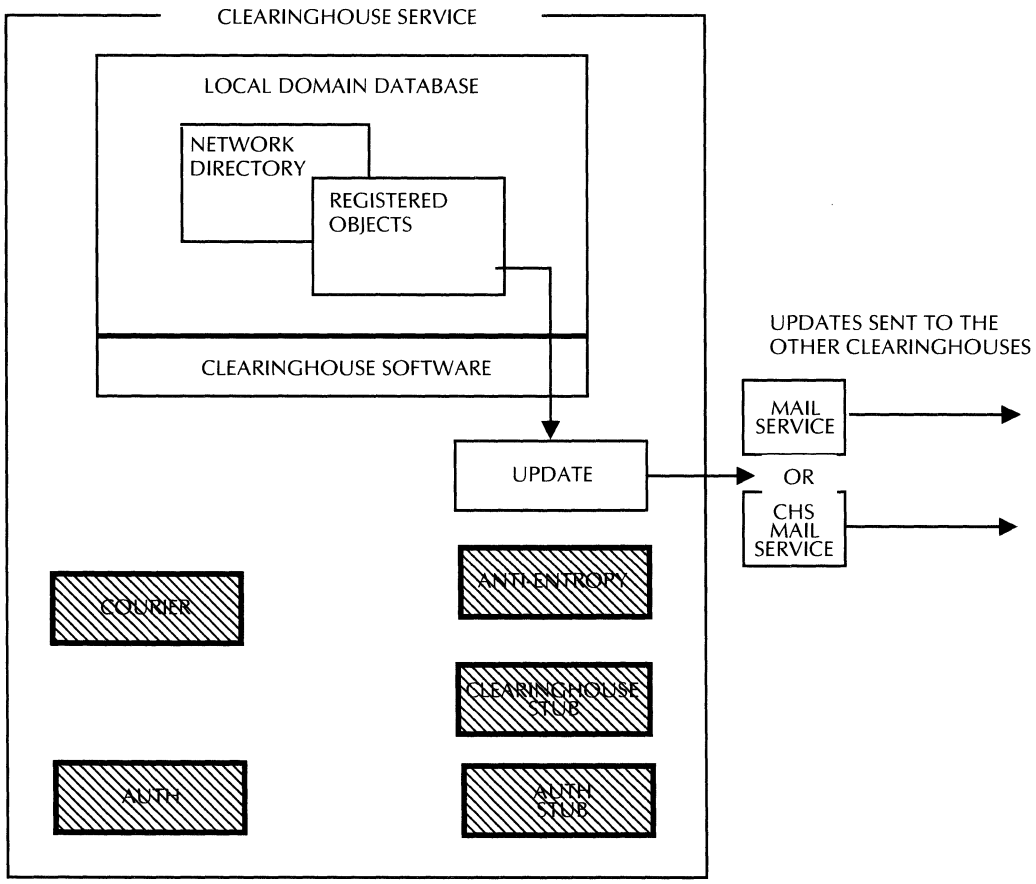
receives an update to delete a nonexistent database object, no action occurs. If it receives an update to add an already existing object, it operates on the update as though it were a request to change the properties of the object.

As illustrated in Figure 3–9, immediately after a client makes a change to the database, the Clearinghouse Service update software generates a message describing those changes. The message is in the form of a mail note.

The Clearinghouse Service sends the message to the coresident Mail Service. If there is no Mail Service running with the Clearinghouse Service, the Clearinghouse Service creates its own mail service, one that is only used to send and receive updates.

The updates are sent to all the Clearinghouse Services that contain copies of the domain. Updates received from other Clearinghouse Services follow a converse path. The Clearinghouse Service modifies the local domain database according to the information received in the updates.

Figure 3-9. **Clearinghouse database update**

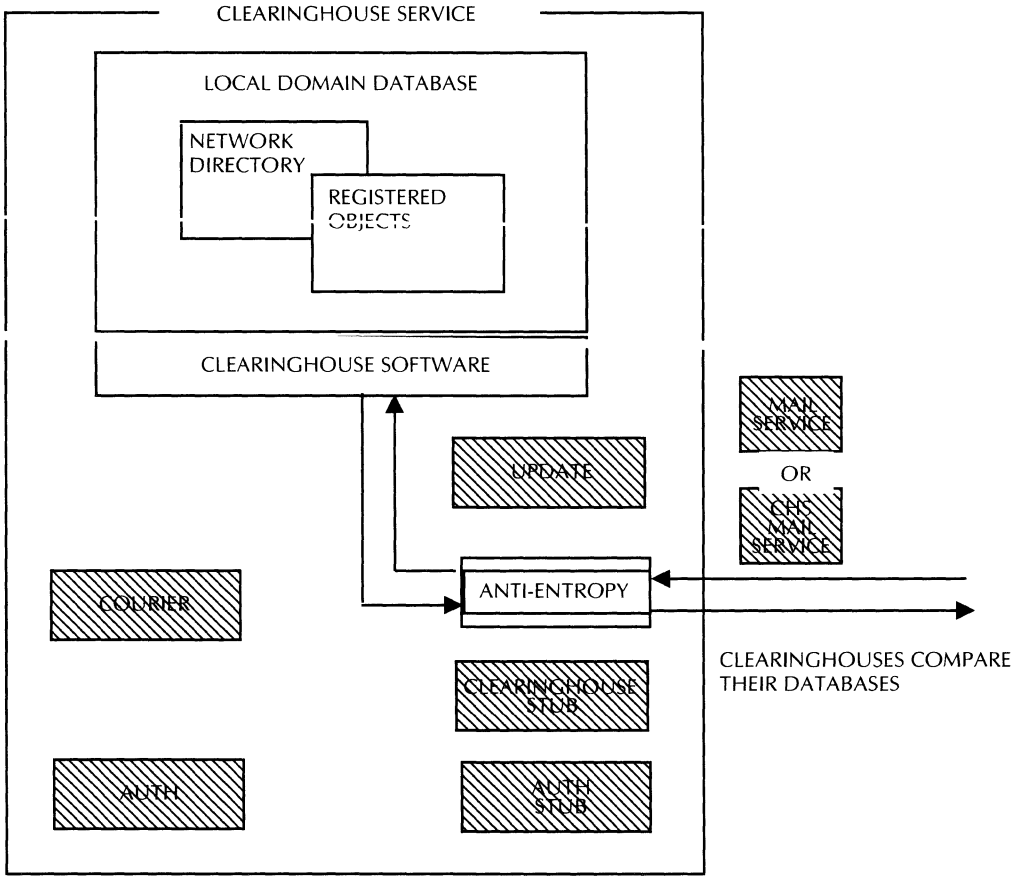


Database consistency checks

At periodic intervals the Clearinghouse database compares the accuracy of the local domain database with all the databases that contain copies of the domain. This differs from the update messages, since consistency checks are done through the network and on a real–time basis. Nightly checks are done in segments; an entire database is not corrected each night.

As illustrated in Figure 3–10, the anti–entropy software checks the database for recent changes. If changes are found, the software connects with the other Clearinghouse Services in the network. Update messages to the databases are compared. Any conflicts or discrepancies in database information are resolved using the information contained in the most recent update messages.

Figure 3-10. **Clearinghouse consistency checks**



Client authentication

The Authentication Service provides the distributed network with a way to identify and validate network clients. The Authentication Service uses the Clearinghouse Service as its database. The identity of a client or service is contained as a Clearinghouse distinguished name. There is a secret key associated with each name. Knowledge of the secret key is necessary to claim that identity.

Authentication Service Components

Authentication stub—Allows clients to access the Authentication Service. The stub may interact with multiple authentication servers to perform client validation.

Authentication database—The Authentication Service uses the Clearinghouse Service database. All data needed for client validation is stored on the Clearinghouse database. You use the Clearinghouse commands to start, stop, and expunge the Authentication Service.

Authentication Service—The distributed service supplied by a set of cooperating authentication severs. The Authentication Service provides a standard secure communication protocol to network clients. The service provides credentials and verifiers, and procedures for creating and verifying them.

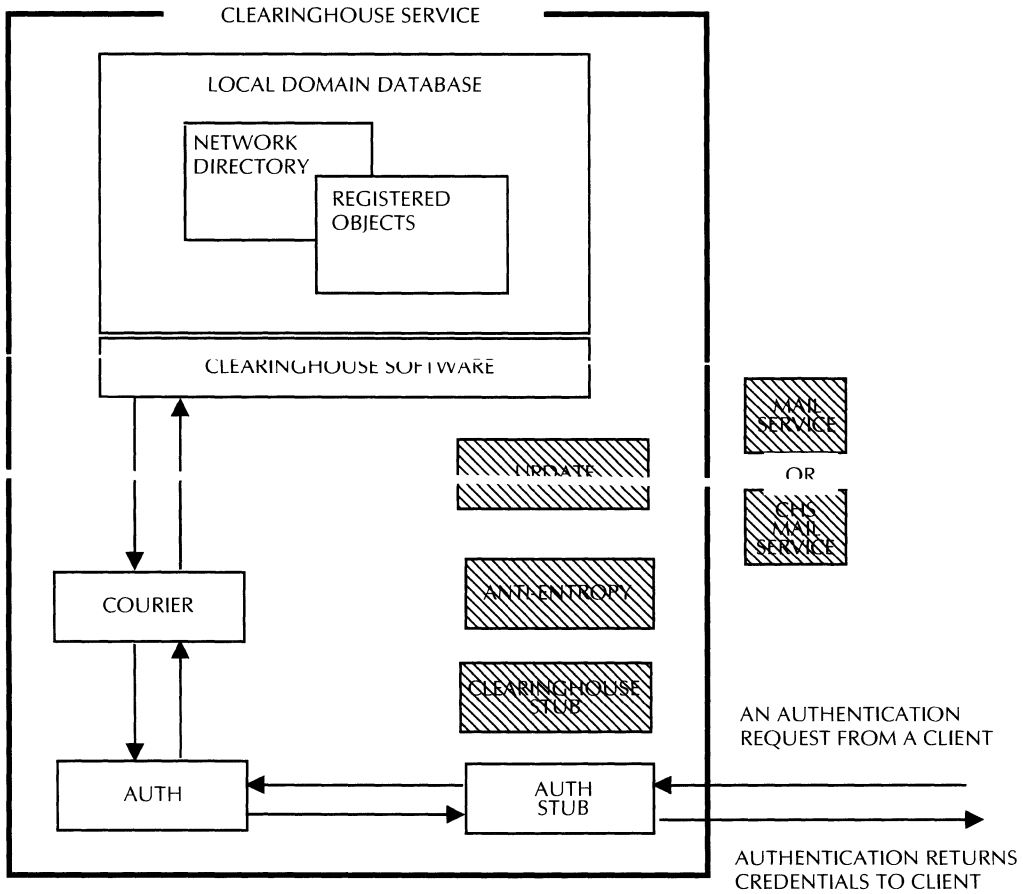
Communicating clients enforce the Authentication protocol. Each client is responsible for obtaining credentials and verifiers, sending them with each message, and checking them on receipt of each message.

Authentication process

As illustrated in Figure 3–11, a network client makes an authentication request to the Authentication stub. The stub passes the request to the Authentication software, then to the Clearinghouse software as a Courier remote procedure, and finally to the database.

The Clearinghosue software searches the database for the object. When found, the software sends the information back to the Authentication software where it is processed. It is then sent to the Authentication stub and finally it is returned to the requesting client.

Figure 3-11. **Authentication process**



Passwords

A password is a confidential and unique alphanumeric word that is created and used by each user to verify, identify, and gain access to network services.

Passwords are like the combination to a safe, therefore they should be chosen carefully. It is recommended that a password be at least 12 characters long and not something that would be easily associated with the user.

The correct combination of registered user name and registered user password authorizes an individual to the network access privileges associated with that name.

Keys

A key is a numerical string derived from a password. The authentication mechanism uses the National Bureau of Standards Data Encryption Standard (DES) encryption keys to convert a password to a numeric key.

A key is the information Authentication uses to validate the identity of a client. Although each user has only one password, they all have two keys: a strong key and a simple key. The strong key is used on machines capable of implementing a strong authentication scheme, and the simple key is used on machines that are not capable of strong authentication.

Levels

To accommodate the varying security needs and capabilities of all system residents, two levels of authentication are available: strong and simple.

The authentication process is flexible enough to allow both highly complex devices and relatively unsophisticated devices onto the network. An unsophisticated device is one that does not have the resources or complexity to implement complex protocols or encryption algorithms.

Strong authentication—Services that use strong authentication rely on a complex encryption method to encode credentials.

With strong authentication, the initiator contacts the Authentication Service and receives the credentials for the client. Credentials contain data that is encrypted in such a way that only the intended recipient can understand it. The client presents the credentials when it calls the recipient.

Contained within the credentials is the identity of the initiator and a special conversation key. The Authentication Service generates the conversation key, encrypts it, and return it to the initiator. Thereafter, each message that flows between the initiator and the recipient includes a unique sequence number that is composed of the system time and the recipients processor ID, encrypted with the conversation key. This encrypted number is called the verifier.

Strong credentials have an expiration date. The expiration date is set by the Authentication Service, and is typically between a few hours and a few days. A set of credentials may be used in any number of calls to the recipient until the credentials expire. A verifier can be used only once. A new verifier must be computed for each message sent to a recipient. A verifier expires shortly after it is sent.

Simple authentication—There is no encryption of data when using simple authentication. Passwords and verifiers are exchanged in simple text. The client sends a name and password to the Authentication Service, and authentication returns a simple verification.

For simple authentication, credentials and verifiers are passed in the same manner that they are for strong authentication, with two major differences: nothing is encrypted and the credentials and verifier are created without the aid of the Authentication Service. The credentials are the name of the initiator, and the verifier is the initiator's simple key.

To validate the credentials and verifier the recipient contacts the Authentication Service. The service checks whether the verifier (simple key) matches the credentials (initiator's name).

Terminology

Clearinghouse Service database — Specialized database used by the Clearinghouse Service to store the names and properties of network objects. The Clearinghouse database should not be confused with general purpose databases that are designed to support management information systems.

client — Software that submits requests to a service on behalf of a user.

courier — XNS protocol that defines the format of request or reply information between system elements.

database replication — Copying a domain database onto a remote Clearinghouse Service.

expanding ring broadcast — Algorithm used to locate an available or answering system or service within a network. An expanding ring broadcast begins when a client makes a BroadcastforServers call to the local network. The client waits a reasonable amount of time for a reply, then repeats the broadcast. If no replies are received, the client makes a BroadcastforServers call on a network that is one hop away from the client. The client increases the number of hops by one until a reply is received or the limit (6) is reached.

hop — Unit of measure used for data moving through a network. Each router or gateway data must transverse is considered a hop.

pattern — Three–part name that includes an asterisk (*) as a wildcard character.

property — Value or sequence of values attached to a database object.

protocol — Rules used to govern network transactions.

registered object — Network object that is registered in the Clearinghouse database. A registered object has a name and properties associated with that name.

resource Service or device attached to a network.

server Generic term for the hardware on which services run.

service The generic term used for software which provides a service to clients.

stub Software that functions as an agent for the Clearinghouse and Authentication services. A stub may interact with numerous Clearinghouse Services to perform a function for a client.

System Administrator User assigned the duty of maintaining a network service. System Administrators have privileges that permit them to create, modify, or delete objects in the database.

user Person registered as a user in the Clearinghouse database.

4. Clearinghouse protocols

Several standard protocols perform fundamental roles in the Clearinghouse architecture. There is also a data format standard present that defines the structure of the entries stored in the Clearinghouse database. Combined, these standards define both the client–to–Clearinghouse interface and the connection of the distributed Clearinghouse System.

Protocol list

Authentication Protocol

The Authentication Protocol gives clients credentials that are used to verify their identities when accessing various network services, including the Clearinghouse itself.

There are two variants of credentials; strong, a tamper proof encryption based form of security, and simple, a less secure form. Many application protocols require credentials to identify a user and enable access controls.

The Authentication Protocol also defines operations for updating the Clearinghouse's authentication data and changing passwords.

Bulk Data Transfer Protocol

The protocol used to transfer the contents of large data items.

Clearinghouse Access Protocol

The Clearinghouse Access Control Protocol allows clients to inspect and change the access controls on a database.

Clearinghouse Entry Format Standard

The Clearinghouse Entry Format Standard defines the property sets associated with the objects, such as File Services and users registered in the Clearinghouse database. The format allows additional object types to be defined without changing the underlying Clearinghouse.

The Entry Format Standard is essentially a contract between the creators of entries, such as a self registering File Service, and the clients of entries, such as workstations looking up File Services. Only certain commands, like [Add User], are aware of the internal format of database entries.

Clearinghouse Interservice Protocol

The Clearinghouse Interservice Protocol provides operations for coordinating the activity of the various Clearinghouse Services that make up the total Clearinghouse System. The operations include inter server domain replication and consistency checks on domain copies.

Clearinghouse Protocol

The Clearinghouse Protocol defines the basic access operations for performing database queries and updates. The protocol provides the operations used to locate and list objects, and retrieve and modify object properties.

The Clearinghouse Protocol, based on Courier, forms a Courier remote program. In addition to the normal Courier procedure calls, the Clearinghouse uses an expedited call facility that allows simple operations to be performed via packet exchange without incurring the cost of network streams.

Clearinghouse Update Protocol

The Clearinghouse Update Protocol is used to propagate database updates among multiple servers. Unlike most NS

protocols, the Update Protocol uses electronic mail, rather than network streams and Courier, as its transport medium. This method of propagation ensures reliable updating even when Clearinghouse Servers are down and unable to receive the update in real time.

Courier Remote Procedure Call Protocol The Courier Remote Procedure Call protocol defines a single request/reply or transaction discipline for an open–ended set of higher–level application protocols. Courier standardizes the format of request and reply messages.

Time Protocol The protocol used to supply the correct time to the Clearinghouse and Authentication Services.

5. Clearinghouse remote errors

When Clearinghouse and Authentication remote procedures fail, the services report the nature of the failure to the client. Each error type references a category of conditions. The error arguments define the problem.

Error list

The following types of errors may occur during an attempted Clearinghouse operation:

- argument errors
- authentication errors
- call errors by either the Clearinghouse Service or the Authentication Service.
- property errors
- update errors
- wrong server errors

The argument errors for each of these error types are discussed in this section.

Argument error

The Clearinghouse reports the error, "ArgumentError," if the service cannot complete an operation because of an error in the arguments of a remote procedure.

There are two types of argument errors:

- An illegal combination of parameters, in which case the service does not attempt to execute the procedure.
- A bad specification, in which case the service attempts to execute the procedure, but no valid object exists.

There are several argument errors:

illegalProperty—The property is not available to the client.

illegalOrganizationName—The argument is incorrect, too long, or contains a wildcard character.

illegalDomainName—The argument is incorrect, too long, or contains a wildcard character.

illegalObjectName—The argument is incorrect, too long, or contains a wildcard character.

noSuchOrganization—The organization name does not exist.

noSuchDomain—The domain name does not exist.

noSuchObject—The object name does not exist.

Authentication error

The Clearinghouse reports the error, "AuthenticationError," when a server refuses to perform an operation because a client's credentials are incorrect or invalid.

There are two types of authentication errors;

- An authentication error because the Authentication Service determined that the client lacks the credentials required to perform the specified operation.
- A calling error can occur due to lack of service resources, lack of information, communication failure, or a server crash. When a calling error occurs, the Authentication Service generates an authentication CallError.

There are several authentication error arguments.

credentialsExpired—A set of credentials has timed out.

credentialsInvalid—The credentials are unacceptable to a service. If the credentials are strong, the decrypted credentials have the wrong format. If the credentials are simple, the Authentication procedure could not locate the name in the database.

inappropriateCredentials—Incorrect credentials were used for a remote procedure; strong credentials were submitted for a simple credentials operations, or vice–versa.

other—An undefined error occurred.

proxyExpired—The proxy has timed out. The original initiator sets the proxy expiration time.

proxyInvalid—The proxy is unacceptable. Only strong proxies can be submitted to the Authentication Service. The error means the decrypted proxy has the wrong format, or a simple proxy was sent to the Authentication Service.

verifierExpired—The strong verifier has timed out. Verifiers expire within a few minutes.

verifierInvalid—The verifier was unacceptable to a service. If the verifier is strong, the decrypted verifier has the wrong format. If the verifier is simple, the simple key (as presented in the database) did not match the verifier.

verifierReused—A recipient has already used this strong verifier. Verifiers are invalid after the first use.

Call error--Authentication Service

The Authentication Service reports the error, "CallError," when it rejects a client operation due to lack of service resources or server malfunction.

There are several call error arguments.

accessRightsInsufficient—The client does not have the proper access rights to request an operation.

badKey—A client attempts to add a key with bad parity bits to the database.

badName—A client has used an invalid Clearinghouse name to create a key operation.

databaseFull—The Authentication database has no more room to store data.

domainForNewKeyUnavailable—A client attempts a **createStrongKey** or a **createSimpleKey** operation, and the server where the new key would be stored is unavailable.

domainForNewKeyUnknown—A client attempts a **createStrongKey** or a **createSimpleKey** operation, but the server where the new key would be stored is unknown.

keysUnavailable—The required Authentication Service is unavailable.

other—An undefined error occurred.

simpleKeyAlreadyRegistered—A client attempts a **createSimpleKey** operation, but the key already exists.

simpleKeyDoesNotExist—The key necessary to generate a set of simple credentials does not exist in the database.

strongKeyAlreadyRegistered—A client attempts a **createStrongKey** operation, but the key already exists.

strongKeyDoesNotExist—The key necessary to generate a set of strong credentials does not exist in the database.

tooBusy—The service is processing other operations and cannot process another one at this time.

Call error--Clearinghouse

The Clearinghouse reports the error, "CallError," when it rejects a client operation due to a lack of service resources or server malfunction.

There are several call error arguments.

accessRightsInsufficient—The client does not have the proper access rights to request an operation.

other—An undefined error occurred.

serverDown—The service cannot complete an operation because another Clearinghouse Service must be accessed, and that service is unavailable at this time.

tooBusy—The service is processing other operations and cannot process another one at this time.

useCourier—The expedited procedure call protocol cannot be used for this operation. The client should retry the operation using Courier.

Property error

The Clearinghouse reports the error, "PropertyError," when an operation fails due to an error associated with the property argument of a procedure.

There are several property error arguments.

distinguishedObject—Lists the name of the object associated with the property error.

missing—The specified Clearinghouse object has no property value.

wrongType—The specified Clearinghouse object has a property value, but it is the wrong property type; item instead of group, or group instead of item.

Update error

The Clearinghouse reports the error, "UpdateError," when a database modification procedure (add, delete, or change) fails.

There are several update error arguments.

databaseOverflow—There is not enough room in the database to store the modification.

noChange—The modification procedure did not change the database.

objectOverflow—Too much data is associated with the object being modified.

outOfDate—The data to be modified has a more recent time stamp than the time stamp of the client's operation. The data is more recent than the request to modify it.

Wrong server error

The Clearinghouse reports the error, "WrongServer," when it rejects a client request because the service does not contain the portion of the database the client wants to access.

6. Clearinghouse references

We referenced the following documents to create this section of the *GLOBALVIEW Document Services Technical Reference Manual*:

Authentication Protocol XNSS 098605 – May 1986

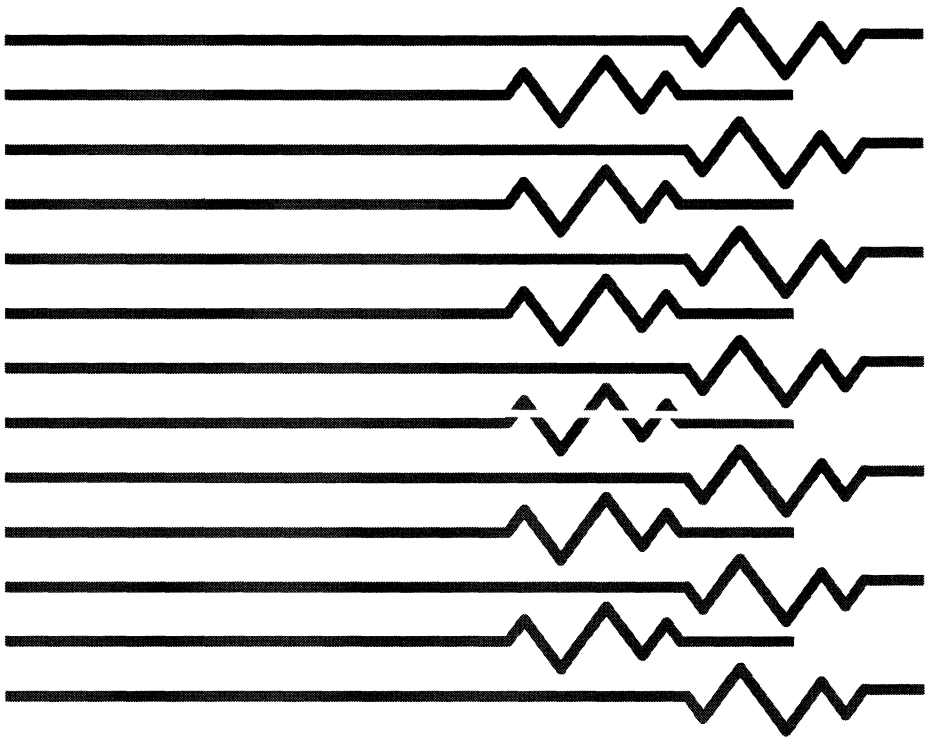
Clearinghouse Entry Formats XSIS 168404 – April 1984

Clearinghouse Protocol XNSS 078404 – April 1984

Courier:The Remote Call Protocol XSIS 038112 – December 1981

The Clearinghouse: A Decentralized Agent for Locating Named Objects in a Distributed Environment OPD–Y8103 – October 1981

Mail Service



7. Mail Service overview

This chapter describes the Mail Service, which consists of a collection of programs which implement the Xerox Mailing Protocols Standard.

Introduction

All Xerox Mail Services on an internet combine to form a single mail transport system capable of supporting the exchange of electronic mail among the users of that internet and with users of other Mail Systems. This distributed Mail System provides communication through the cooperative interaction of an expandable set of Mail Services which work together to provide an integrated facility supporting the posting, transport, and delivery of electronic mail. The Mail Services run on servers and are accessed by clients using the appropriate protocols.

The protocols provide for the following:

- Posting of mail using delivery slots
- Delivery of mail using delivery slots
- Medium-term storage of mail in Mail Service mailboxes
- Full data transparency, allowing the transport of arbitrary data from the originator to the recipient using the Mail System.
- A standard message format for interpersonal messages, facilitating a basic level of end-to-end compatibility among Mail System clients.

This chapter discusses the mailing protocols and related data formats used for interaction between clients and the Mail System.

These protocols are application level protocols, which use several other protocols. Requests to a Mail Service are communicated using request-reply or the transaction discipline of remote procedure calls.

The Mail Service is a message transfer agent for Mail Stubs, the User Agent and client portion of the software. A message is defined as a unit of electronic mail consisting of two parts, the envelope and the content. The message envelope contains information that is necessary to route the message to its eventual destination and, once delivered, to interpret the message content.

The Mail Service relies heavily on the Clearinghouse Service. Each potential message recipient must be registered in the Clearinghouse. Also, each potential recipient must have a mailbox on one of the Mail Services. A mailbox is a logical container which is the end repository for messages destined for a given recipient within the local mail transport system. Each

mailbox is associated with only one fully qualified recipient name and can be identified either by that name or an alias. To complete delivery of a message, the CHS database must contain the mailbox location associated with the message recipient. Also, the server that contains the Mail Service must also be registered with the CHS. Clearinghouse registration is performed by the Mail Service when a mailbox is created for a user and by the installation procedures when a Mail Service is brought online.

Principles of operation

A service is an entity, a combination of software and hardware that accepts and responds to submitted requests for some type of service. A Mail Service is composed of one or more Mail Services working cooperatively to form an integrated facility. A client of the Mail System is an entity that submits requests to services that make up the system. All interaction between the Mail Service and the client is initiated by the client. A Mail Service never initiates activity with a client.

This section describes the Mail System. It explains what the Mail System is and how it works. For information on System Administrator duties refer to the *System Administration Guide*.

The Mail System provides facilities for network clients to send and receive messages across a network. A message can consist of a mail note or other objects such as files, folders, or reference icons.

Any network user who is registered with the Clearinghouse can send mail messages. Any registered user who has a mailbox on the network can receive messages. Users need a mailbox to receive messages, but not to send them.

Users can access the Mail Service from any workstation running GLOBALVIEW or VIEWPOINT.

System components, as well as users, can utilize the Mail System to communicate with each other. For example, multiple Clearinghouses on a network notify each other about database updates by sending mail messages to each other.

If a network has a gateway to other networks, users can send mail messages to mailboxes on other networks. Similarly, users with mailboxes can receive messages from users on other networks.

When users send messages to recipients on the same network, they only need to know the registered name of the recipients. The routing of the message through the Mail System is transparent to the sender and the recipients. If a user sends a message to a user on a different network, they may need to know the recipient's fully-qualified (three-part) name.

Although an electronic mail message can contain contents in any format or any language, not all types of workstations can read all formats. For example, non-GLOBALVIEW applications can read mail notes created in GLOBALVIEW, but may not be able to read attached documents created in GLOBALVIEW.

The Mail Service maintains the privacy of the contents of a mail message. Only the sender and recipient of a message can view the contents of the message.

How the Mail System works

The Mail System is implemented by multiple Mail Services distributed through a local area network. As the number of users growS, more Mail Services can be added to a network.

The various Mail Services located across a network automatically and actively cooperate to form a unified Mail System. The main benefits of the distributed approach are increased availability and modular growth. If one Mail Service stops functioning, another Mail Service can pick up mail that would normally be picked up by the failed Mail Service.

Like a local post office, each Mail Service provides three basic functions:

- Accepts mail posted by nearby senders
- Holds mail awaiting pickup by nearby recipients
- Forwards mail destined for nonlocal recipients either to the Mail Service where the recipient's mailbox resides, or to a Gateway Mail Service if the message is destined for users on a different network.

GVDS does not have a Mail Gateway. However, the GVDS runs compatibly with the 11.0 and later versions of XNS servers on Xerox hardware. So although you cannot install a GVDS Mail Service gateway on a Sun, you can connect a Xerox server to your network, and install a Mail Service gateway on it. The GVDS Mail System can then forward mail for external recipients to the Mail Service gateway.

Each Mail Service has a database that holds mailboxes and maintains lists of queues, such as the queue of mail messages to be processed.

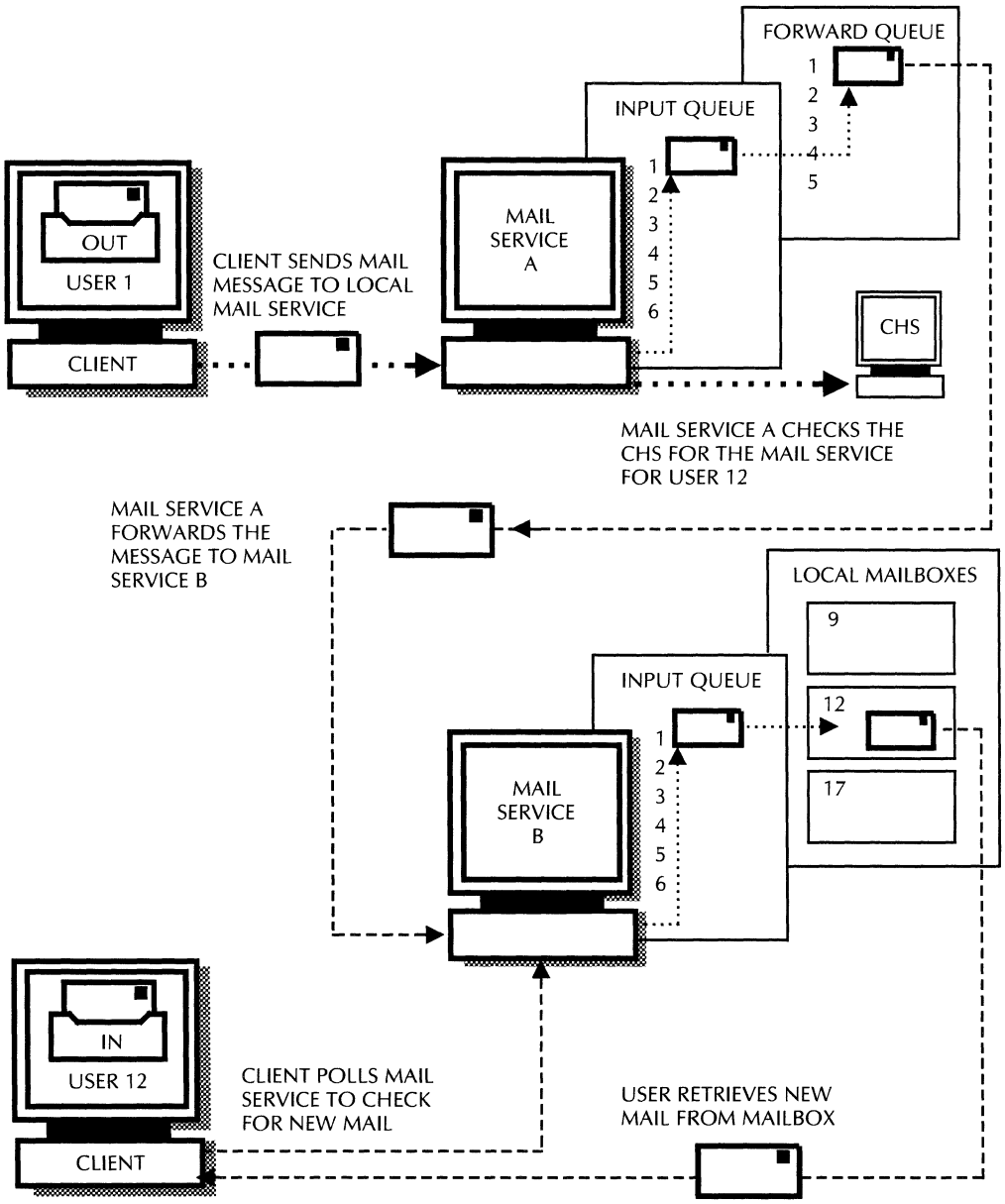
Each user may have up to two mailboxes. If a Mail Service which contains a mailbox for a user stops functioning, the Mail Service which contains the second mailbox for that user can store their incoming mail.

For a workstation to interact with the Mail System, it must contain a Mail Stub. A Mail Stub is a low-level software package that provides an interface to the Mail System. In GLOBALVIEW, the Mail Stub uses the mailbox icon to pass incoming messages, and the Outbasket icon to receive messages to be sent.

How the software works

Figure 7-1 shows a representation of how an example mail message travels from the sender's workstation to the recipient who has a mailbox on a different Mail Service, or a nonlocal recipient.

Figure 7-1. **Mail to nonlocal recipient**



When a user sends a mail message from a workstation, the Mail Stub on that workstation must first locate the correct Mail Service to which the message should be sent.

The Mail Stub uses an "expanding ring broadcast" search to locate the appropriate Mail Service. When a message arrives, the Mail Service adds a postmark to it, indicating the time and the name of the server. It then places the message in its input queue, which is a first-in-first-out queue. When the message

reaches the top of the input queue, the Mail Service determines where to send it to next.

There are four types of recipients:

- Local recipients -- users who have mailboxes on the local Mail Service. A logical copy of the message is placed in the local recipient mailbox. The term logical copy means that only a single physical copy of the message exists, and the copy is shared by all mailboxes on the server.
- Remote XNS recipients -- users who have their mailboxes on other Mail Services on the same local area network. Messages for remote recipients are placed in the forwarding queue and are forwarded to the appropriate server.
- Disjoint XNS and foreign recipients -- users whose mailbox resides on a remote XNS network. Foreign recipients are users whose mailboxes are contained on remote nonXNS networks. Mail can be sent to disjoint and foreign recipients only from a network that contains a Mail Service Gateway installed on a Xerox server or other hardware that supports Mail Service Gateways.

  Messages for disjoint and foreign recipients are sent to the Mail Service gateway. Delivery to a disjoint XNS user takes place through an XNS Gateway while delivery to a foreign recipient occurs through a gateway to another mail protocol such as CCITT's X.400 or TCP's SMTP.
- Distribution lists -- groups of users who may be any combination of local recipients, remote XNS recipients, or disjoint XNS and foreign recipients. Each of their members are added to the recipient list of the message, and then each member is processed individually.

Sending mail

If the mailbox for the recipient of a message is on a different Mail Service or network, the message must be forwarded to the relevant Mail Service or gateway.

The Mail Service makes one attempt to forward each message in its forwarding queue. If forwarding does not succeed, the Mail Service places the message in its pending queue.

Each Mail Service maintains a pending queue of messages that need to be forwarded but for which a previous forwarding attempt has failed. The Mail Service periodically attempts to process these messages again. Currently these attempts are made about once every two and a half hours. If the message times out while in the pending queue, the Mail Service generates a nondelivery report and deletes the message. A message times out when the period of time since it was posted exceeds the time out setting in the Mail Service software.

When a message reaches the Mail Service containing the recipient's mailbox, the message enters that service's input queue. When it reaches the top of the input queue, the Mail Service delivers it to the recipient's mailbox, where it remains until the recipient retrieves it.

Message reports

Sometimes the Mail Service cannot successfully deliver a message. Reasons for nondelivery include the following:

- The message timed out in the pending queue, perhaps because the network is down, or the destination Mail Service is too busy, too full, or down.
- The recipient became invalid after the message was sent.
- The message was sent to a distribution list, and one of the members of that list was invalid.

If a message exceeds the time out setting, or if its recipient became invalid after the message was sent, the Mail Service stops trying to deliver the message and generates a nondelivery report.

In most cases, the nondelivery report is sent to the originator of the undelivered message. The exception is when the reason for nondelivery is an invalid name in a distribution list. In this case a special nondelivery report is sent to the owner of the distribution list.

If all the return attempts are unsuccessful and if the sender selected [Alternate recipient allowed] on the mail note's property sheet, the report is placed in the local postmaster mailbox. The report is discarded if there is no postmaster mailbox. If the sender did not select [Alternate recipient allowed], the report is discarded.

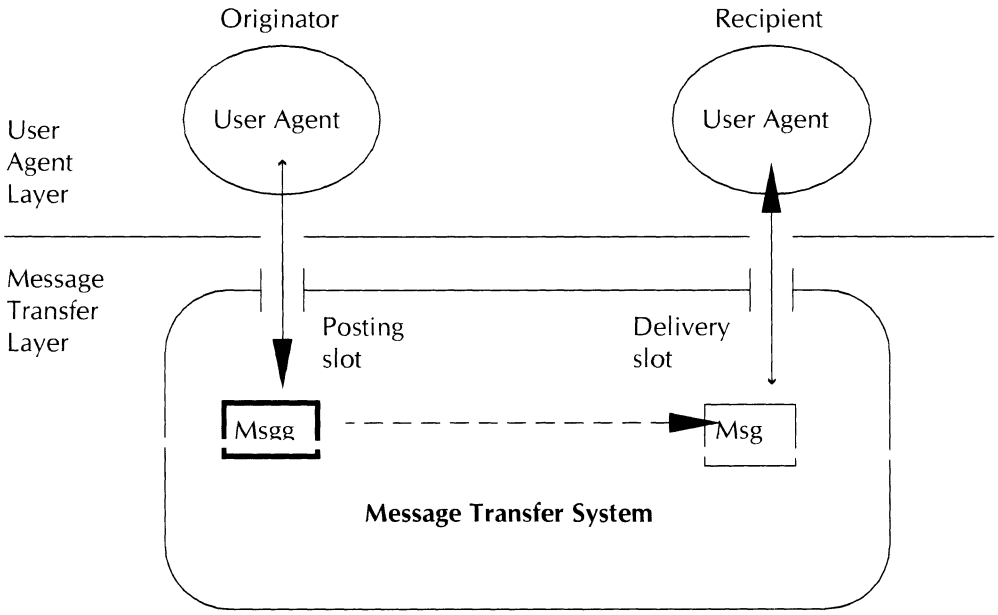
Components and service design

This section describes the Mail Service components and design, primarily the Xerox Mailing Protocols Standard. This standard includes several protocols used by the Mail Service.

Mail System architectural model

Figure 7-2 illustrates the Mail System Architectural Model, a framework for understanding the components of the Mail Service. This illustration provides a visual representation of Mail Service function and interaction.

Figure 7-2. **Basic X.400-based architecture**



In this model, the fundamental boundary is drawn between the User Agent layer and the Message Transfer layer. The User Agent layer includes all facilities for original composition and posting of messages by the originator, and for later delivery and processing of messages by a recipient. Actual transport of the message from the originator's User Agent to that of the recipient is performed by the Message Transfer layer.

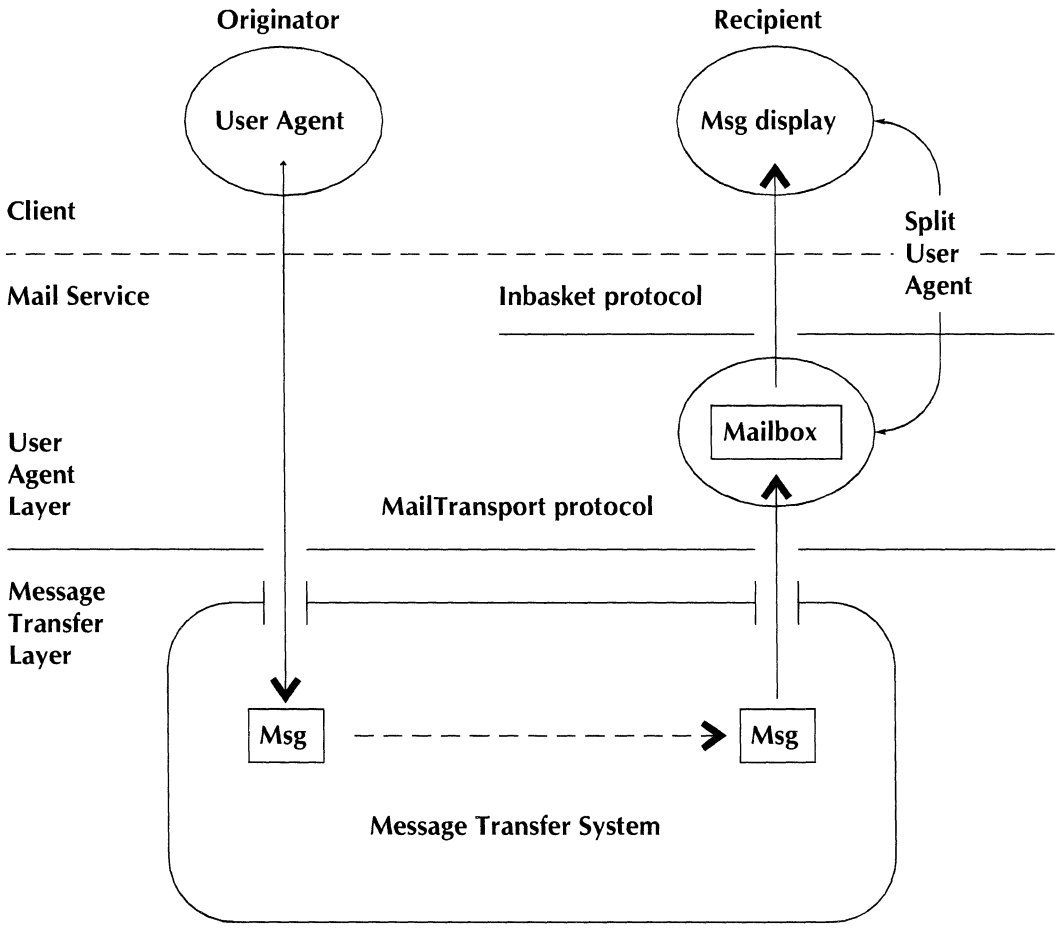
In terms of this basic model, the Mail Transport protocol defines the interaction between the message transfer layer and the User Agent layer and provides operations for sending and receiving mail using the posting and delivery slots. Accessing the Mail Service according to this protocol thus equates the physical boundary between the client and the service with the interaction between the two layers. Using the Mail Transport protocol for delivery implies that mail is transferred sequentially to the client using the first-in-first-out process.

The Mail Transport protocol provides roughly the same functionality as the X.411 of the CCITT X.400 series of recommendations. However, the Mail Transport protocol specifies that the passage of mail is always initiated by the client. This differs from X.400, where the service is allowed to "push" mail through to the client.

Sequential access to messages is acceptable for some clients. However, some users may want to access their mailbox from several workstations. Another protocol called Inbasket allows this.

The Inbasket protocol provides random access to messages in a message repository, or in a mailbox residing on the Mail Service. The Inbasket protocol corresponds architecturally to an internal interface of the User Agent layer. In Figure 7-3, the boundary between the client and the Mail Service is the Mail Transport protocol used to post messages, and the Inbasket protocol used to retrieve them. The function that is architecturally assigned to the recipient's User Agent is split between the workstation and the service as shown.

Figure 7-3. **Extended architectural model**



Procedures and operations

This section provides in-depth detail about the primary Mail Service protocols.

Mail Transport protocol

The Mail Transport protocol is the fundamental protocol for submission and delivery of messages. It is essentially a Courier program exported by all Mail Services and used by clients for origination and receipt of mail.

Posting and Retrieval of messages using slots

The Mail Transport protocol defines posting slots and delivery slots to support originators and recipients of messages. Using the posting and delivery operations described below, clients cause messages to flow through these slots. The passage of a message through a posting slot implies that the Mail System will deliver the message efficiently. Similarly, the passage of a message through a delivery slot implies that the receiving client has accepted responsibility for that message, and that the Mail Service has delivered the message.

Each Mail Service contains a single posting slot. Posting slots are anonymous, in the sense that mail addressed to any recipient can be posted to any slot. To post a message, a client locates a posting slot on a Mail Service and then invokes the posting operation of the Mail Transport protocol. The selection of a posting slot involves two steps:

- Locating a set of available Mail Services.
- Choosing which one of them to use.

A client can locate one or more nearby Mail Services by executing an expanded ring broadcast. Every Mail Service on a given net will respond by returning a package of useful information.

Authentication

Many Mail Transport remote procedures require the client to verify their right to access the service. The qualifying information is composed of two items, credentials and a verifier, as defined in the Authentication protocol.

These two parameters identify the client to the service and contain enough information to validate the client's right to perform the requested operation. The credentials need not be strong, and the names used in them should be those of the user and the Mail Service. In some of the delivery operations, the credentials are not passed each time, because they are validated at the time the session is established and are valid for a sequence of operations.

A Mail Service may refuse to perform an operation if the client's credentials or verifier are invalid or not sufficient for that operation.

Inbasket protocol

The Inbasket protocol is an important enhancement to the basic functionality provided by the Mail Service's Mail Transport protocol. It is intended to enhance the functionality of the system when used for the transport of messages. Specifically, it allows a given user to use any number of different User Agents to access incoming mail by providing a medium term repository for delivered mail that resides within the Mail Service and can be accessed by multiple clients.

Delivery slots

The lowest level component of the User Agent resides in the Mail Service, which constantly polls the delivery slot and immediately takes delivery of any messages that arrive. Messages are stored in a mailbox in the order delivered. The other components of the User Agent reside on client machines and can access the mailbox to inspect messages held there. Unlike the delivery slot, the mailbox does not require that each message be removed from the server before the recipient can inspect the contents. Instead, a set of random access operations is provided for selective listing, retrieval, and deletion of messages in the mailbox.

In most cases, the sharing of a given mailbox among multiple clients tends to occur sequentially in time, but this is not required. Multiple clients can access the same mailbox concurrently.

Each mailbox is associated with a specific recipient name.

Message identification

Each message in a mailbox is identified by a numeric index, which is a unique identifier for the message in the mailbox. Indices are permanently assigned to messages; that is, the index of a message does not change across sessions. Indices are assigned as a function of time. The Mail Service can determine whether one message was placed into the mailbox before another by examining the indices of the messages.

A contiguous set of messages is identified by a range of indices in operations that apply to multiple messages.

Message status

The message status indicates whether the user has been made aware of the presence of a message, as mediated by the User Agent. Because the high level components of the User Agent that effect reception reside in the client, the mailbox facility only provides a supporting mechanism to record the status of each message. The status must be kept up-to-date in the client. The status is designed to reflect the normal progress of a message through the following two stages that occur following delivery:

- When the message is new, the user has not been made aware of the presence specifically. Typically, the User Agent will provide some global signal alerting the user to the presence of one or more messages in the mailbox.

- When the message is known, the user has been presented with a summary description of the message, often in the context of a larger listing of some or all the messages in the mailbox. Typically, the User Agent will include in this summary some indication that the message has not yet been viewed in full, such as placing as asterisk (*) before the message summary.

It is important to note that the status of a message represents the extent to which the user has been made aware of the message. In particular, the transfer of partial or complete information about the message from the Mail Service to the client of the Inbasket protocol represents an internal event within the User Agent and does not constitute a change in the messages's logical status.

A user defined status is also provided for sophisticated User Agents. The user defined status is not interpreted by the Mail Service. One application that might make use of this status is a program that indicates messages of high importance.

Inbasket remote procedures

Each Mail Service implements the set of mailbox operations as a Courier remote program.

Remote errors—When a remote procedure completes successfully, it returns results as specified in the definition of the procedure. However, conditions can arise before or during execution of a procedure that make successful completion of the procedure impossible. For example, a client may have specified incorrect arguments in a remote procedure call, or a required resource may be unavailable.

When such conditions occur, an error is reported to communicate to the client the nature of the problem. Each error encompasses an entire class of possible conditions and the specific problem is further described by the arguments of the error. For example, AccessError indicates that the Mail Service rejected the requested operation because of an access problem.

The following types of mailbox errors can occur:

- Authentication – The Mail Service is unable to perform a mailbox operation because the client's credentials are invalid or not sufficient for that operation.
- Access Error – Even if the caller's credentials are valid, the caller may not be allowed to invoke the requested operation because of the lack of access privileges or the state of the service.
- Session Error – An operation may be rejected because of incorrect session-related data.
- Service errors – An operation may be rejected because of lack of service resources or server failure.
- Index Errors – An operation may be rejected because a specified index does not identify a message or body part.
- Inbasket in use error – Although the current Inbasket protocol allows clients to access the mailbox concurrently, older versions do not.
- Other errors – In unusual cases, an operation may be rejected because of an error other than the types listed

above. For example, an expedited procedure call was made to a nonexpedited procedure, or the number of message body parts does not match the message's table of contents.

Expedited procedure calls

The Mailing Protocols utilize the Packet Exchange Protocol defined in the Internet Transport Protocols to call some Mail Service remote procedures. Remote procedure calls made in this way are called Expedited procedure calls. An Expedited procedure call is frequently more efficient than the corresponding Courier call. The following characteristics of a remote procedure call make the use of such a low-level protocol appropriate:

- The calls are usually confined to the local network.
- The amount of data allows the argument and result list to be less than the maximum Packet Exchange size.
- Minimum delays are important for performance reasons.
- The operation may be repeated many times with no ill effects.

Because of the restriction on the amount of data that can be conveyed in a single packet, not all Mail Service operations can be invoked as expedited calls. However, for every operation that can be expedited, the function operates in exactly the same way as the corresponding Courier operation; it produces the same effects, returns the same results, and reports the same errors.

Terminology

authentication Verification of user identification and access rights based on the information in the Clearinghouse database.

Clearinghouse Service XNS service that provides a central repository for names, addresses, and properties of system resources, permitting XNS system elements (workstations, servers, and so forth) to locate them.

client Workstation or logical process that initiates some service and that benefits from the service having been provided.

courier XNS protocol permitting the initiation and control of remote processes, including the transfer of information and control parameters associated with such processes.

datagram Unit of transmission in certain communication systems, over which individual routing and accounting control is exercised, and which is a constituent building block of a message.

host Computer that is accessed by users.

Inbasket protocol In XNS, the protocol used to obtain messages received in a user's mailbox on a Mail Service.

mailbox In XNS, the place where a user's mail is kept before it is retrieved by the user. Generally identified by a user's name and the name of the Mail Service where the user is registered.

mail format Xerox standard for the content of a mail message (aligned with CCITT X.420P2 specification).

Mail Service Service providing mail facilities to clients on a network.

mail transport protocol In XNS, the protocol for posting messages to the Mail System.

protocol Set of rules agreed to by two communicating parties for accomplishing a specific set of tasks.

remote procedure call protocol *See* Courier.

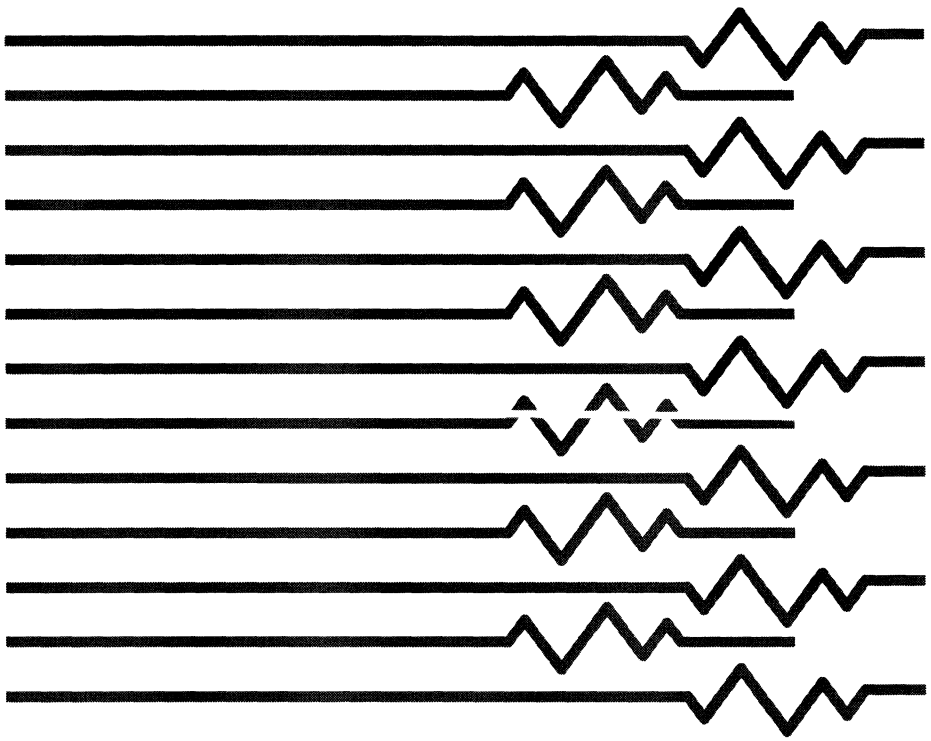
server Combination of hardware and software capable of performing a set of services.

service (1) Set of tasks performed on behalf of a client

(2) Set of software resources which implement a number of XNS protocols to achieve a result on behalf of a client.

socket Abstract location in an XNS system element that can originate or receive communication.

Print Service



8. Print Service overview

This chapter shows you in technical detail how the Print Service works. Refer to the *System Administration Reference* for information about the user interface and user procedures and commands.

Introduction

The Print Service is a set of applications which run on GLOBALVIEW on the Sun platform and provide printing resources to a local printer or shared printer in a networked configuration. The Print Service is designed as an integrated system.

Overview

The Print Service provides Interpress 3.0 and PostScript printing with NeWSPrint capabilities to network citizens. The Print Service processes print jobs, responds to status requests from network clients, and responds to user commands through the Service Executive.

The typical print job is processed by the Print Service in three steps. The Print Service first receives the print request from the network client. The job is then spooled to the local file system, and finally inserted into a queue with other jobs.

Jobs are processed in the order received and according to the priority specified by the client. When a job reaches the front of a queue, the Print Service identifies the attached printer, decomposes each page into a raster image, and transmits the raster image bitmap to the printer.

The Print Service supports the Sun SPARCprinter and the HP LaserJet III printer.

Print Service design

Version 1.0 of the Print Service is based on an open architecture design that allows the following:

- Rapid introduction of printer options to support new printers and output devices
- Architectural support for new Print Service features that exceed the capabilities of the architecture used by earlier Print Service products.

New printing options

The new Print Service open architecture uses an object-oriented modular building block approach that allows for rapid introduction of new printing options. Many of the software components that make up network printing software are designed as common elements that can be shared by various printing options without modification. The Interpress decomposer and queueing facilities are examples of such components. Reusable building block objects and components make it easier for the independent and rapid development of printing options.

Architectural support of new features

Some of the requirements that exceed the capabilities of previous architectures which are designed into the Print Service open architecture include the following:

- Support for multiple network protocols and user interfaces
- Support of complex network protocols such as the XNS Printing Protocol version 5, or DPA
- Secure printing
- Priority queuing.

Components

The Print Service contains the following logical parts:

- Transaction request clients
- Core
- Processing services.

The core has the following subsystems:

- Transaction request handler
- Job control
- Job processing.

{ADD INTRODUCTORY TEXT.}

- Client input
- Transaction request (core)
- Job control (core)
- Job processing (core)
- Processing services (output)
- Processing objects (output).

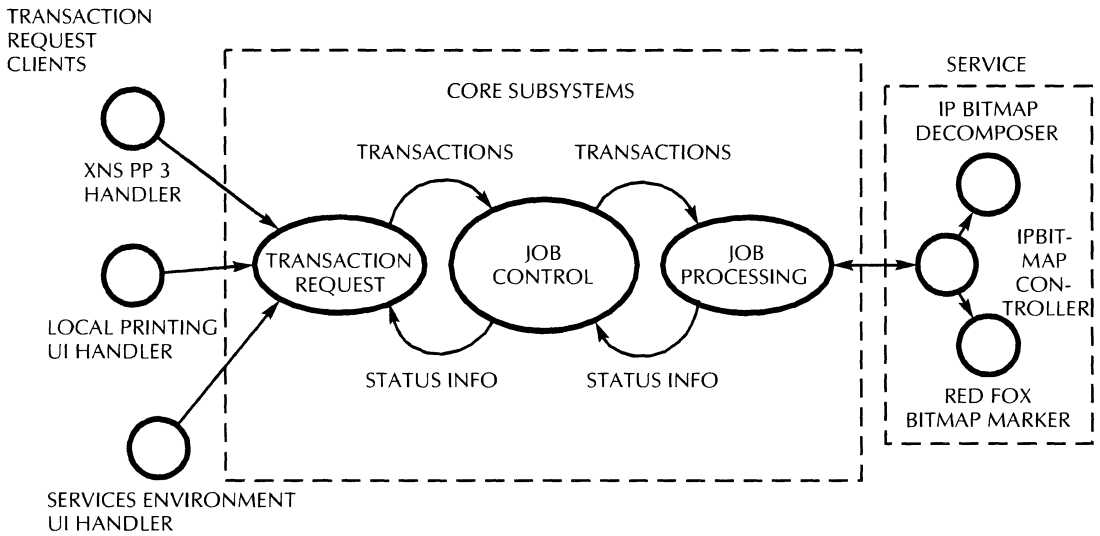
Clients

A client initiates requests for Print Services. A client can be a network user or software acting for a user. A client requests service for users or for other software clients from the Print Service. A client can request a system value change, information, or a print job.

Core subsystems

Figure 8-1 shows the subsystems that form the core of the Print Service open architecture. The core subsystems are common to all configurations of Print Services. The core is designed to be independent of protocols, user interfaces, and printing options. As a result, the core should not change as new protocols, user interfaces, printers, and other output devices are added.

Figure 8-1. **Print Service components**



The core contains these major subsystems:

- Transaction request handler
- Job control
- Job processing.

Each subsystem is covered in detail in the following sections.

Core overview

The transaction request subsystem processes all client requests for the Print Service. It is the entry point for external systems into the Print Service core. It interfaces with transaction request handlers, and the Print Service components. The handlers are modules written for specific network protocols, user interface platforms, or formats. To add a new client, you must design a handler for that client. Handlers are included in this release for these clients:

- XNS PP3 Protocol Handler
- Local Printing Interface (Services Environment-based)
- Network Printing User Interface (Services Environment).

The transaction request subsystem primarily services transactions initiated by the handlers. It also relays status and property information from Print Services to protocol and user interface handlers and back to the client. The transaction request subsystem does not service transactions directly, but requests the job control subsystem to service the clients.

A special transaction requires the Print Service to process (print) a document file. This transaction results in a job for the Print Service. A job consists of a document file to process and any information about how the job should be completed.

The job control subsystem queues and schedules jobs for actual processing. The job control subsystem also maintains and revises the queues. Thejob control subsystem includes these components:

- Print queue
- Job scheduler
- Status handler
- Resource manager.

The job processing subsystem manages the processing of the job. Job processing does not actually process the job, but it dispatches a job that it receives from job control to an appropriate service. As the job is being processed, the job processing subsystem may report status information it receives from the processing service back to job control. Job control in turn may act on this information or pass it to the transaction request subsystem.

Basic work flow

Work flows in one direction through the Print Service, from left to right, as shown in Figure 8-1. The client is to the left; the service is to the right. Each subsystem is a client of the subsystem on its right and a service to the subsystem on its left.

The Print Service is a passive resource acting only in response to transaction requests from a client. External clients ask the transaction request subsystem to start work. Work flows to the right, passing through the downstream subsystems and arriving at the destination (printer).

In some cases, work does not reach the job processing subsystem and the transaction is handled by an intermediate subsystem. For example, a request for an operation on the queue does not have to go all the way to the job processing subsystem since it can be handled by job control. Requests that are fulfilled by an intermediary process are not passed on to the next subsystem.

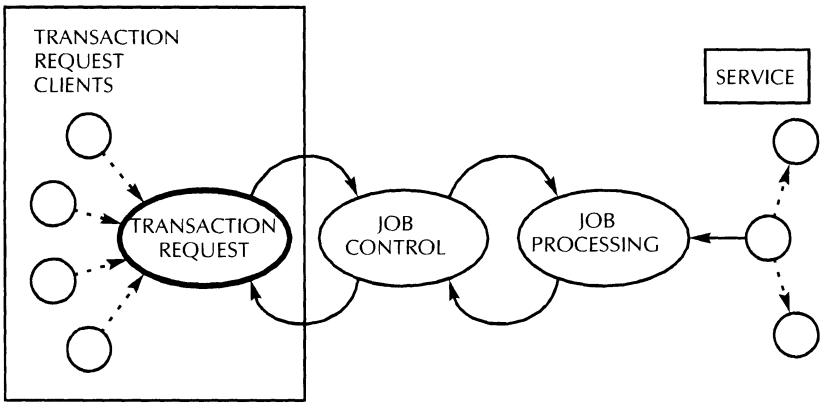
Responses to information and status requests flow in the opposite direction, from right to left. A subsystem may report asynchronous events to a subsystem upstream. For example, a printer may report a paper jam. This information is sent back to the client. Subsystems report status information to client subsystems using call-back procedures.

How the transaction request subsystem works

The transaction request subsystem handles interactions with all external clients. Clients are network users or software acting on behalf of a user. If the client is a user, then the user is interacting through a user interface. If the client is software, then the software is passing on a request which it received from a user or other software. Everything a client receives from the Print Service comes from, or through, the transaction request subsystem.

Figure 8-2 shows how the transaction request subsystem interacts with the other core subsystems snd services.

Figure 8-2. **Transaction request subsytem**



A client can make three types of requests from the Print Service:

- Action – request to change a system value, such as the break page frequency, default substitution font, and so forth.
- Information – request to read and report system values
- Print Services – request to process a job (print a hard copy of a document).

As the handler of client interactions, the transaction request subsystem captures client requests and packages them into a syntax and form that the rest of the system can process. The transaction request subsystem also monitors significant changes in the system, and notifies clients with pending requests.

The transaction request subsystem is entirely transaction oriented. It does not initiate any activity, it only responds to client requests and system status reports.

The transaction request subsystem responds synchronously and asynchronously to clients. Synchronous responses can be values for client show status requests. Asynchronous responses are messages about system conditions which are broadcast to the entire system. Not all client handlers can receive asynchronous broadcast messages.

Client support

The transaction request subsystem supports a variety of clients, including different user interface paradigms and platforms, and different network protocols. The transaction request subsystem provides a generalized interface and data format that is independent of any particular client protocol context. The client handles the context-specific details and translates data into the general transaction request subsystem format. The transaction request subsystem can support multiple clients from a single configuration.

Typically, the client acts as the middleman between its own client and the transaction request subsystem with its generalized core format, translating data as necessary and handling context-specific details. For example, a client may be a GLOBALVIEW local printing application handler that maps user actions from the GLOBALVIEW context to the transaction request subsystem format. Other clients can include handlers for specific protocols such as XNS or TCP/IP, or a batch job processor. Any hardware or software that can request service from the system can be a client of the transaction request subsystem.

The client's functionality depends on the requirements and restrictions in its environment, platform, or context. For example, a handler for XNS Printing Protocol version 3 (PP3) cannot set the default substitution font. The transaction request subsystem does not require a minimum functionality; the requirement is set by each client.

Typical work flows

When a user request first arrives at the Print Service system, it appears as a registered client of the transaction request subsystem in the handler. The handler gathers the required information and packages it into the standard core format required by the Print Service core. The transaction request subsystem then accepts the client request in the well-defined format for processing.

If the transaction request subsystem cannot process the request entirely by itself, it passes the request to the rest of the system. If the request requires an action or response from another subsystem downstream, the transaction request subsystem will wait for the result, and then pass the result back to its client.

If the user request is a print request, the transaction request subsystem passes the request to the rest of the system and returns the message, "jobReceived," to the user. The transaction request subsystem does not process any part of the print request; it passes the request to the rest of the system for processing.

When a system condition message is returned from the system, the transaction request subsystem evaluates it, decides which clients to notify, and then notifies them. Clients must explicitly tell the transaction request subsystem which conditions and messages to pass to them. The client must register a *statusProc* with the transaction request subsystem at startup if it wants to be notified asynchronously of system conditions. A client can also specify which events it wants to be notified of. The transaction request subsystem will only notify clients of those events which they have registered for.

Job control

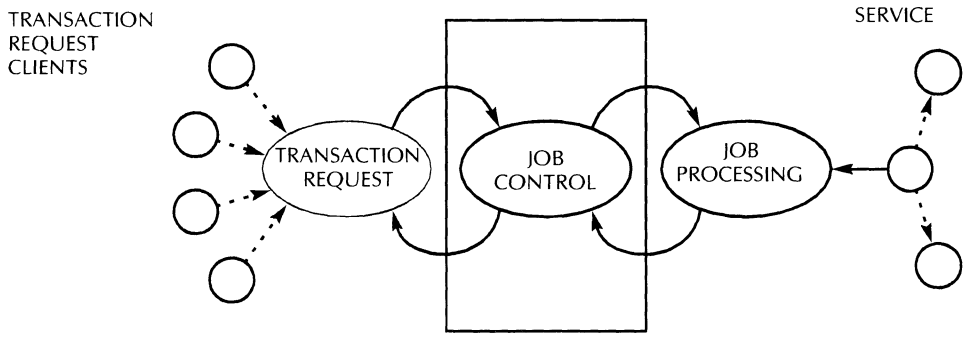
The job control subsystem is the point at which all print jobs are retained while waiting for further processing. Job control manages all print jobs that have completed spooling. Job control handles routes, schedules print jobs to job processing, and handles status messages returning from job processing. Status messages can arrive synchronously, asynchronously, or both. Job control sends all messages that it cannot process upstream to

other client subsystems that may be able to understand the message and respond.

Figure 8-3 shows how the job control subsystem interacts with the other core subsystems snd services.

The job control subsystem works with jobs according to their status and characteristics. Job control also provides the mechanism to access runtime data from other subsystems, and the global information that is maintained by the environment subsystem or job control itself.

Figure 8-3. **Job Control Subsystem**



The job control subsystem has the following major components:

- Print queue
- Job scheduler
- Status handler
- Resource manager.

Print queue

The print queue is the heart of the job control subsystem. Once a print job has been spooled successfully, the job and all related information is passed from the transaction request subsystem to the job control subsystem. Job control then stores the job, control data, attributes, and the job control packet.

Each print queue element is assigned a unique ID that is used by all entities in the system to identify it. The print queue is the only subsystem that can directly access all queue elements to store and retrieve information.

The print queue is a dynamic structure that can be as large or small as the situation demands. You can also explicitly set the size of the print queue.

A print job goes through several processing stages in the queue. At each stage it is given a status marker. The processing stages and related status markers are as follows:

- Spooling
- Holding
- Pending
- Inprogress
- Complete
- Forwarded
- Aborted
- Canceled.

Print queue elements with the same status markers are linked to become a single queue processing stage. These groups are individual but coresident logical queues (also referred to as *virtual queues*). When the status of a print job changes, the corresponding print queue element moves from one logical queue to another

The print queue maintains and manipulates data regarding its elements. It can also back itself up to prevent loss of content from a failure that requires a system reboot. During system restart, the print queue can restore itself from backup data.

The print queue supports priority queueing and secure printing.

Job scheduler

The job scheduler selects the print job to process after the current job completes. Depending on the print job characteristics, the current availability, and the capabilities of the job processing module, the job scheduler does one of the following:

- Submits the print job to job processing
- Requeues the print job to a different queue stage (where it may need special attention to print)
- Does nothing until a necessary condition arises
- Selects the next print job to process that requires different resources from those used by the current processing job. This option is available only if if the job processing subsystem supports multiple services.

The job scheduler creates and destroys a job record for every job it submits to job processing. This job record contains job-related information for the processing stages.

The job scheduler also provides extra facilities such as the special timer process which starts other time-dependent tasks.

Status handler

The status handler is the only component of the job control subsystem that monitors what occurs within job processing. The status handler provides callbacks for job processing to report the status of the current job and service. This information is reported synchronously or asynchronously.

Although all status information passes through the status handler, it evaluates only the information for which it can take action. If it cannot take action, it returns the service status upstream to the subsystem that initiated communication with the job control subsystem.

The status handler can directly access the job control print queue and update the processing job status.

Resource manager

The resource manager controls job control resources by performing the following tasks:

- Providing spooling space.

- Allocating print queue elements.
- Helping expand or reduce print queue space. You can explicitly set the queue size in the user interface.

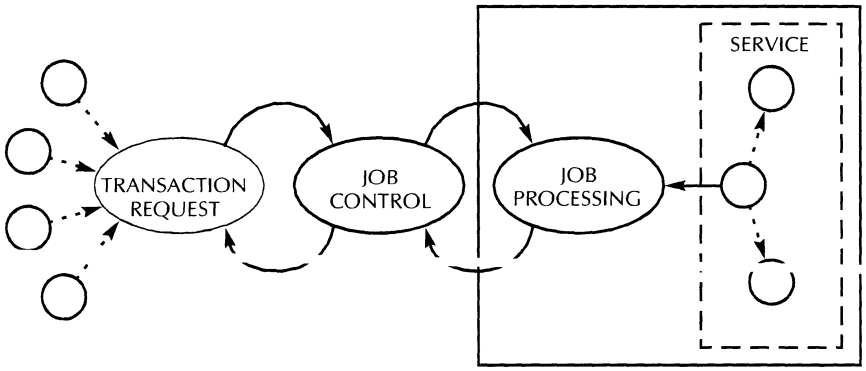
As each job completes, the resource manager returns as much space to the system as possible. Space allocated to the components of the print job being processed, such as the job record, job attributes, and job instructions, is reclaimed when the job is completed. The resource manager links all job-related components, dynamic and static, to the print queue element for each spooled job to back up and restore the print queue.

Job processing subsystem

The job processing subsystem processes the jobs submitted by job control. Although the actual processing of each job is done by one of the service modules, the job processing subsystem oversees the job from start to finish and reports status information during the process.

Figure 8-4 shows how the job process subsystem interacts with the other core subsystems snd services.

Figure 8-4. **Job Processing subsystem**



The job processing subsystem allows job control to cancel, suspend, and resume jobs. It also performs the following tasks:

- Dispatches transactions to other services
- Channels communications between services and job control
- Allocates resources
- Filters status messages from services to job control.

Services

Jobs received by the job processing subsystem are performed by separate entities called services. Each service represents a printing option or set of printing options. For example, one service is for printing on a SPARCprinter.

In addition to processing jobs, each service maintains information about the printing options, including status and properties. Some transactions will retrieve and modify the status information.

The concept of separate services is the basis for the development of separate printing options. Services register themselves separately with the job processing subsystem at

startup. You can add new printing options without affecting or interfering with other services.

Each implementation of a service is composed of software components called processing objects. Figure 8-1 shows the processing objects called IPBitmapDecomposer, IPBitmapController, and RedFoxBitmapMarker.

Processing objects can be reused as modules or objects, or copied by other services. This object-oriented modular building block approach makes it easier to quickly introduce new printing options.

You can reuse or copy the processing objects included in this release of the Print Service to develop different services.

Each service module represents a printing option or set of printing options. The service module is the unit that the system refers to when submitting transactions to printing options. Although the initial release supports one service, the Print Service architecture can support many services at once.

Each service must register with the job processing subsystem by providing a unique ID number and a list of call-back procedures that will perform various transactions. Transactions received by the job processing subsystem are usually directed to a specific service using the unique ID, and the appropriate call-back procedure is used to complete the job.

Service modules process the individual jobs. Each service must complete the job, and report when the job is either finished or the system determines that it cannot be finished. The service may provide status information during the job.

Each service also keeps status and property information. Some transactions retrieve and modify state information. These transactions are not associated with a particular job and can occur any time after the service has registered itself. A service, therefore, is active during job processing and at all times after registration.

Processing object output

A service is composed of software components called processing objects. A processing object represents a unique hardware or software device that performs a certain function. For example, a processing object can interpret Interpress masters and produce bitmaps for each page. Another processing object can take a bitmap and print it on a particular printer. A third processing object can control and coordinate the actions of the first two processing objects. The three processing objects work together to allow the Print Service to complete a job.

These processing objects are included in the Print Service software:

- IPBitmapDecomposer – decomposes Interpress masters into bitmaps one page at a time.
- BitmapMarker – prints bitmap images to a printer one page at a time. Each printer or printer family will have a separate implementation of the bitmap marker. For example, there is one bitmap marker for SPARCprinters and another for HP LaserJet III printers.

- IPSubService – controls Interpress bitmap decomposers and bitmap markers to form a printing option for printing Interpress masters.
- NeWSPrintSubService – submits PostScript jobs to Sun NeWSPrint for processing
- Service – controls IPSubService and NeWSPrintSubService to provide Interpress and NeWSPrint printing.

There are two types of processing objects: control objects and processing objects. The job processing subsystem communicates directly only with control objects. Control objects are a special category of processing objects that manage other processing objects to finish a job. Normally the control object registers the service with the job processing subsystem.

Control objects regulate and control classes of processing objects, not a specific implementation of a processing object. Objects in the same class support the same interface and perform similar functions.

The concept of classes of processing objects makes the building block approach to new printing options possible. A single control object can control different combinations of processing objects if all belong to appropriate classes. For example, you can use a class of processing objects that are markers for specific printer engines to construct different printing options by interchanging the combination of control object, common resource processing objects, and marker processing objects.

Several services can reuse the same processing objects. For example, you can build a variety of services that print to Interpress printers using combinations of processing objects which might include IPBitmapDecomposer and IPSubService. This model permits the rapid development of new printing options.

Work flow

A job is handed off to the job processing subsystem with reference to a service and a list of data and instructions that apply to it. The job processing subsystem invokes the call-back procedure required to accomplish the job and flags its internal tables that the service is in use. The service remains tagged as in use and unavailable for another job until it informs the job processing subsystem that it is finished and available to accept another job. This usually occurs when the preceding job is completed; however, some services can accept several jobs at once.

A job does not need to be directed to a specific service. A client can request a list of services, all of which can do a particular job. The job processing subsystem queries each service about capability and current availability for a job, and passes the information to the client. The client can then choose a service from the list of available and qualified services, or allow the job processing subsystem to choose a service.

The job processing subsystem responds to asynchronous events sent from the services while processing a job. Services report status information during job processing. The job processing subsystem either acts on this information or relays the status back to job control.

Job control can request the job processing subsystem to cancel, suspend, or resume any job. The job processing subsystem then instructs the service responsible for that job to quit, stop temporarily, or resume a process.

Terminology

client Network citizen (for example, a PC or workstation) that sends a job to the Print Service for a user.

core Modules of the Print Service which do not change as new printing options, protocols, or user interfaces are added. The major core subsystems include: the transaction request subsystem, the job control subsystem, and the job processing subsystem.

decomposer Formatter that converts a document file into rasterized bitmap format for printing.

formatter Software component that converts a document file into an intermediate form for processing.

imager Library of high-level graphic procedures designed to create an image suitable for a specified output device. It is typically used by the decomposer software together with the PDL interpreter to create raster images for the printer.

interpreter Software program that interprets Page Description Language (PDL) masters and initiates the calls to the imager to create images of the document pages. The decomposer software uses the interpreter to convert the PDL master into a raster image for the printer.

job Client request to process a document file. Job attributes are sent by the client with the job to define how the system should process the job. Properties associated with a job include the document file, attributes, printing instructions, and status information.

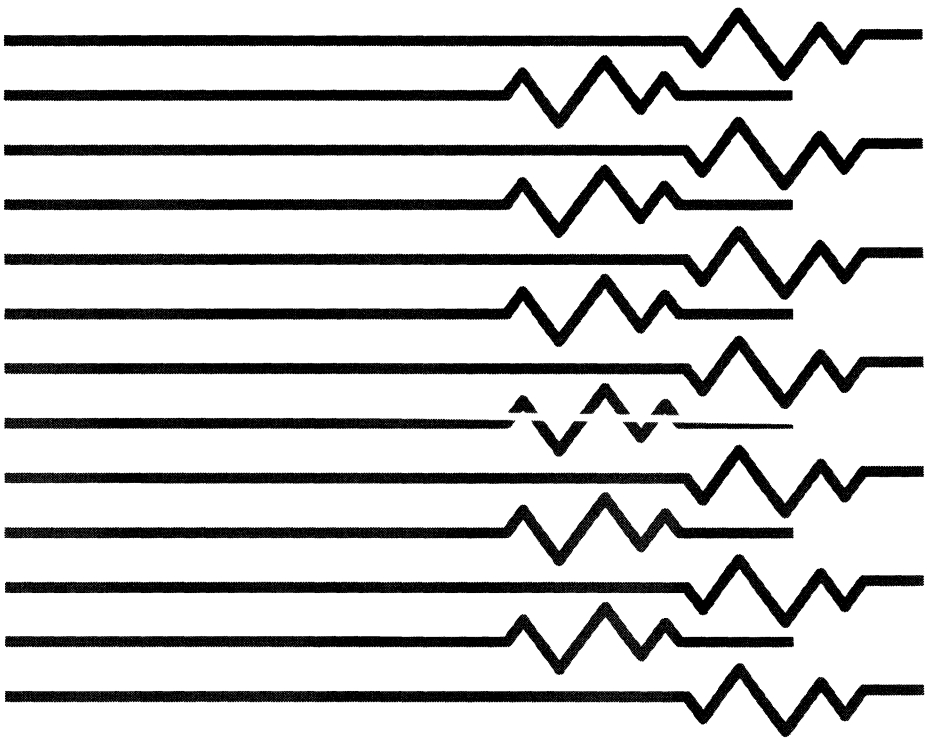
PDL (Page Description Language). Language for encoding images independent of the output medium. Used to describe pages within documents for electronic reproduction. PostScript and Interpress are two PDLs.

printing option Configuration of the Print Service that supports a particular print engine or conversion service. The current implementation of the Print Service can support only one printing option at a time. You can change printing options by modifying the configuration file.

secure printing Print Service feature that holds a print job until the user enters a password to explicitly release it for processing.

spooler Software component that receives a job and places it in the print queue.

File Service



9. File Service overview

This chapter explains the design of the Xerox File Service. The File Service runs as an application within UNIX, and is recognized by SunOS as an application. The File Service has proprietary attachments to the kernel.

Introduction

The File Service provides file storage and retrieval services to clients on an internetwork. The File Service recognizes workstation software as a client. The client operates on behalf of a user or another service.

To alleviate space problems, or to store older files, a user can move files from a workstation to a private file drawer contained on a File Server. You create a private file drawer for each user on the network, and a public file drawer for each group of users.

The File Service allows users to copy files to public access file drawers, which allows network access to that information. UNIX users can also access XNS files provided the XNS System Administrator provides them with the necessary access.

A user can copy their desktop to the File Service, and retrieve it from another workstation which has access to the File Service.

Should a File Server or power failure occur, the File Service provides a scavenging facility which removes inconsistent data when the system is running again.

The File Service organizes remote files in volumes, in a hierarchical order. You name each volume, and the File Service registers each volume in the Clearinghouse Service. The Clearinghouse Service registers the volume names as File Service objects, allowing clients to locate any available File Service which is registered.

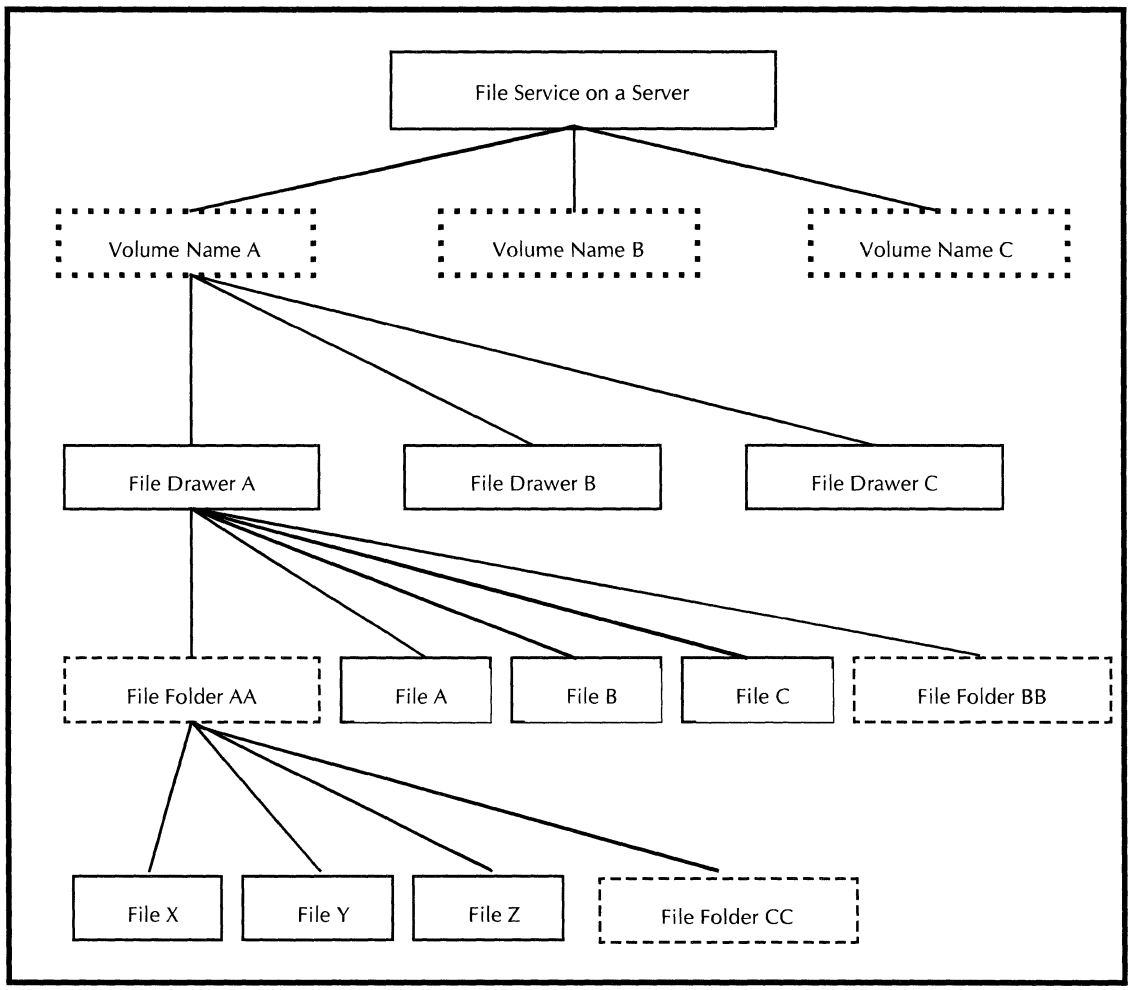
Overview

This chapter describes a Xerox File Service running on a Sun SPARC platform under the UNIX operating system.

The main function of the File Service is to provide a remote storage medium to the network. The File Service allocates storage to users by allowing users to own file drawers with System Administrator-specified storage limits and access rights. The File Service structure is hierarchical; the objects associate with each other in a parent-child relationship. At the top of the hierarchy is the file system root.

Each object in the hierarchy is either a file, a file folder, a file drawer, or a volume. If children are associated with a file, that file is considered a directory, as illustrated in Figure 9-1.

Figure 9-1. **Directory structure**



Requirements and dependencies

The File Service requires the following to operate:

- a Sun SPARCstation or 6500-series Xerox workstation
- SunOS 4.1.1 or greater

The Clearinghouse Service and the Mail Service both use the File Service to backup their databases. The system administrator should do periodic File Service backups. The File Service requires the Clearinghouse Service and Authenication Service to authenticate users and to resolve access lists.

Related publications

To understand how UNIX, NSF, and the UNIX environment operate in a network, see the reference documentation which came with your Sun workstation, or a UNIX reference book. Pay particular attention to information about symbolic links and semaphores.

Volumes

The File Service uses a volume to store file drawers, file folders, and files. A volume is a UNIX directory, and can be one of the following:

- A regular UNIX directory
- A symbolic link to any directory within the UNIX file system (refer to link() in the UNIX Reference Manual)
- A mount point to another UNIX file system
- A mount point to a remote NSF server.

XNS Filing volumes exist in a specific location. You specify this location during installation. The volumes can be local (UNIX directories) or remote (symbolic links).

The File Service has no limit on the number of volumes within a UNIX disk partition. You can also mount volumes as NFile Service devices (refer to the NFS Restrictions chapter).

The collection of files contained by a volume may be made accessible or inaccessible to clients as a unit. To allow network clients to identify the particular volume containing a file of interest, the name of each volume used by the File Service is registered in the Clearinghouse Service, along with the network address of the server on which the File Service is running.

The Clearinghouse Service recognizes the volume as a "File Service name" and not a "File Service volume name," since it is the only File Service entity with which clients need to identify.

The File Service does not require special procedures for primary or secondary volumes, such as Services 12.0. You can make or remove a volume at any time. Clients cannot access the File Server until the first volume exists.

You may configure a server by dedicating a UNIX disk partition for the File Service volumes. This disk partition contains all the data structures that the File Service volumes require.

If the volumes are links to other file systems, the local partition still contains tables and other data for the volume. It is possible that these structures will fill the local partition before the remote file systems are full. To avoid this problem, ensure that the local partition that contains all File Service volumes or links is at least five percent of the remote file system size.

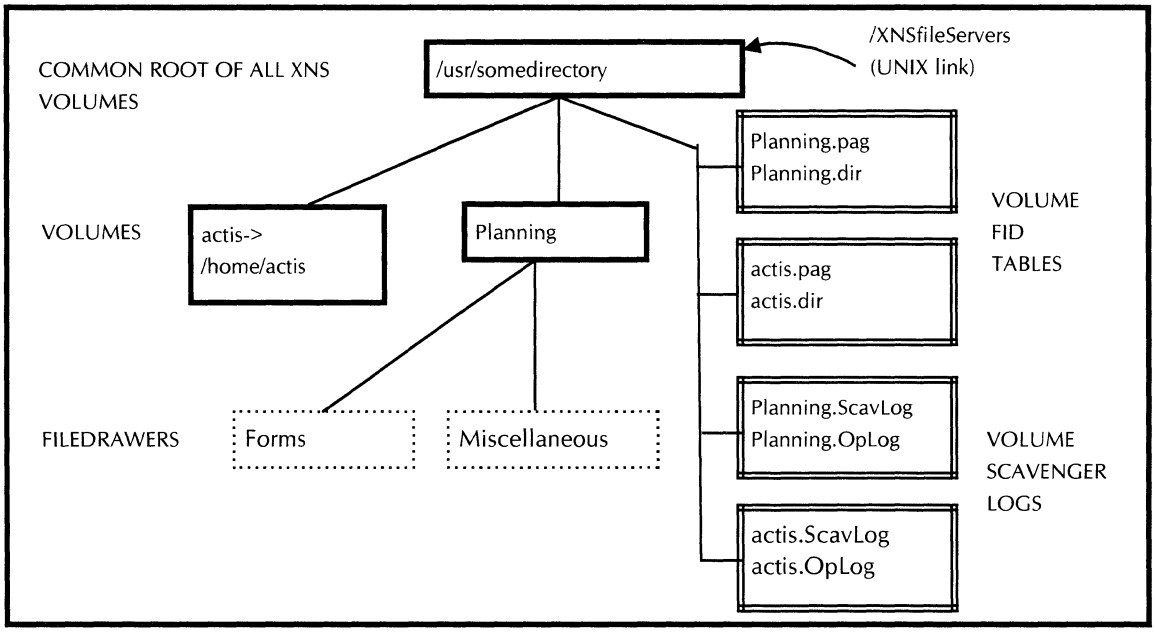
The File Service provides a configurable number of concurrent remote connections; the default is 32. This value is in the configuration file and is set during the File Service installation.

Refer to the "File Service executable files and data" chapter in this document, and in the Release Notes for the location of the configuration file.

UNIX disk partitions

An File Service volume resides on a UNIX disk partition that is specified during installation. Figure 9-2 shows the structure of the volumes named actis and Planning.

Figure 9-2. **Volume structure**



A link (/XNSfileServers) points to the disk partition that holds the volumes. Multiple volumes may be present within this partition. The mount point for this file system is a uniquely named directory.

Volume structure

A volume is a directory in a desired UNIX file system disk partition or directory. A volume can expand to occupy part of, or all of the containing partition. The File Service can use one or more volumes to store file drawers, desktops, file folders, and files.

Volume names stored under the disk partition can be directories or links to other directories.

During installation, you enter the UNIX pathname where all XNS File Service volumes will reside (for example, /disk2). The installation script connects the user-specified location to the /XNSfileServers soft link. The File Service uses the /XNSfileServers internally, so that it can also be a link to any other location in the UNIX file system.

If a client attempts to access a File Service volume whose name is not contained in the /XNSfileServers, the File Service will return the error message, "NoSuchService."

The /XNSfileServers directory contains the FID tables for each volume. The FID tables expand as files are added to the File Service volume. You must insure that this UNIX partition has sufficient space available to accomodate the FID table expansion; otherwise, the clients receive the error message, "VolumeFull."

FID table expansion is 250 to 300 bytes per XNS file. The File Service stores other files under the /XNSfileServers directory for scavenging purposes.

There are no restrictions on the number of logical volumes which can be contained under /XNSfileServers; these are UNIX directories.

Each time you create a new logical volume, the File Service creates the following:

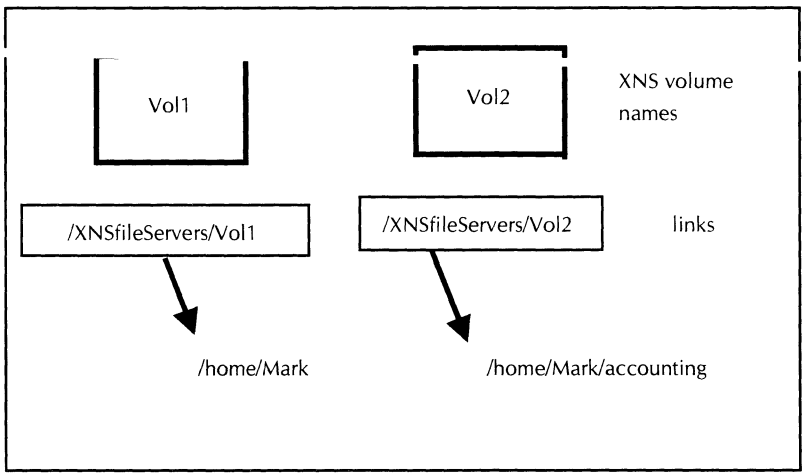
- A UNIX directory or link
- A volume FID table
- A desktop directory within the volume.

The XNS volume name must contain only ASCII characters.

Links

The UNIX link utility allows the File Service the capability to point to a UNIX directory, and use that directory as a File Service volume. As Figure 9-5 illustrates, you can link the File Service volume, Vol1, to the top directory (/home/Mark), and link Vol2 to a decendant directory (/home/Mark/accounting). The File Service considers these links as two volumes.

Figure 9-5. **Links**



Do not allow a volume to be within the hierarchy of another volume.

If a user removes a volume link, the File Service volume linked to the directory becomes inaccessible. For example, if the link /XNSfileServers/Vol1 is deleted, no data is lost, but the files stored on /home/Mark cannot be located.

Volume states

A volume is in either an online or offline state. When a volume is online, it is available to clients. Offline volumes are not available to clients, even if the File Service is started. Only a system administrator may set a volume online or offline. The volume state does not change when the server is rebooted.

All client access to stored files, both local and remote, is through the XNS Filing Protocol exported by the File Service. There are no privileged sessions for local clients (for example, GLOBALVIEW running on the same machine)

File drawers

Each file drawer has an owner on the network. File drawers provide for remote storage of the user's personal files, or of files to be shared among many users. From a workstation, users can store and retrieve files to and from a file drawer.

Client applications may also use a file drawer to store files specific to the application. For example, a file drawer can be associated with a user name registered in the Clearinghouse, which represents the client application rather than a real user.

The total amount of disk storage that a file drawer and the files contained in it can occupy is specified by the page limit attribute of the file drawer. The combined sizes of the file drawer and all the files it contains may not exceed the file drawer's page limit.

Desktop storage

At the request of a user, a desktop can be stored on the File Service. A user may retrieve, list, and delete his or her own desktop. As System Administrator, you may list or delete any desktop.

You can also copy a desktop to tape for backup purposes, since a stored desktop is in a UNIX directory. The File Service does not provide any means to backup or restore desktops or other files. Refer to the UNIX **cpio** or **tar** commands to backup UNIX files.

Sessions

The logon process creates a session, and the logoff process destroys that session. All client filing operations within a single session are serialized. GLOBALVIEW is a client of the File Service, and can establish several sessions to gain concurrency (background operations use one session, and an open file drawer uses a separate session).

Every File Service session requires one UNIX semaphore. The File Service preallocates a group of semaphores equal to the maximum number of allowed sessions. If you increase the session limit, you must increase the number of kernel semaphores.

During the life of a session, the access rights for a particular user

do not change, even if the System Administrator changes the access to files.

For example, if a user attempts to open a file drawer and fails due to insufficient access rights, he or she will continue to have insufficient access even if the System Administrator adds the user's name to the access list of the file drawer. The user must logoff from XGV and log on again to establish a new session. Attempts to access the file drawer during a new session should succeed.

Access

The File Service verifies the access rights of a user by checking each name in the access list until a matching entry is found. If there is a problem while resolving access rights, the File Service returns an error.

Access to files can become more complex when UNIX files are present. Refer to the "UNIX/XNS Access" chapter for information about adding UNIX files or when greater interoperability is necessary.

Creating UNIX filenames from XNS filenames

The File Service uses the XNS filename which is retained within an attribute file, not the UNIX filename.

The File Service uses the following method to create a UNIX filename from an XNS name:

- The File Service does not perform XNS string decoding.
- The File Service changes all characters to 7-bit ASCII (leaving English names alone). The File Service changes unprintable characters such as form feed, newline, and so forth to an ampersand (@).

  The File Service changes a front slash (/) to a backslash (\).
- The File Service truncates the XNS name to the UNIX maximum name length (a compile time parameter, 255 for BSD UNIX , 14 for Sys V).
- The File Service appends an extension consisting of a version number to the resulting UNIX filename. The extension has the form: period, tilde, two digit version number, tilde (for example, .~02~).

The File Service sets and maintains some attributes, such as the file's unique identifier (its file ID,) the date the file was created, the size of the file, and the time of last update to the file. The client sets other attributes, such as the file's name, the files type, and if it is a directory, or data.

File attributes

The File Service uses attributes to describe different properties and controls of files. All data files have an associated attribute file.

File attributes, such as the file ID or filename, identify a file so it can be accessed. Attributes, such as the file's size, its creator, and most extended attributes, provide additional information about the file.

One XNS file is represented by the following UNIX files:

- Data of the file
- One or more attributes in an attribute file

Each client can define and set file attributes specific to the client application, these are extended attributes. When the client stores a file through the File Service, the File Service stores extended attributes with the file, and the client is responsible for maintaining and interpreting these extended attributes.

Different files may have different sets of extended attributes and some files have no extended attributes.

Files within a directory can be ordered based on one of a number of different attributes. They can be ordered, for example, by name or by creation date. Further, access to files can be restricted by the placement of access controls on a file.

When a client requests read access to a file, or to a file's parent, the File Service checks the attributes file, and verifies if the client has access (refer to the "UNIX/XNS Access" chapter).

UNIX retains some attributes within the inode, some are computed from UNIX information, and some are retained within an attribute file.

An attribute file has the same UNIX name as its associated contents file, but resides in the .XNSresources attribute directory (See Figure 9-4). Each directory containing XNS files has its own associated ".XNSresources."

In XNS, a directory may retain data as if the directory were a regular file. Since a directory cannot store user data in UNIX, the File Service creates a ".XNSDirContent" file in each directory to retain the data that associates with the directory (for example, the GLOBALVIEW converter icon, XPIW RES files, and the Emulator icon are directories with attached data).

UNIX files and XNS files can share the same directory.

When performing a change attribute on a UNIX file, the file becomes an XNS file; the File Service creates an attribute file to save changed attributes.

The following sections briefly describe the most common XNS File attributes.

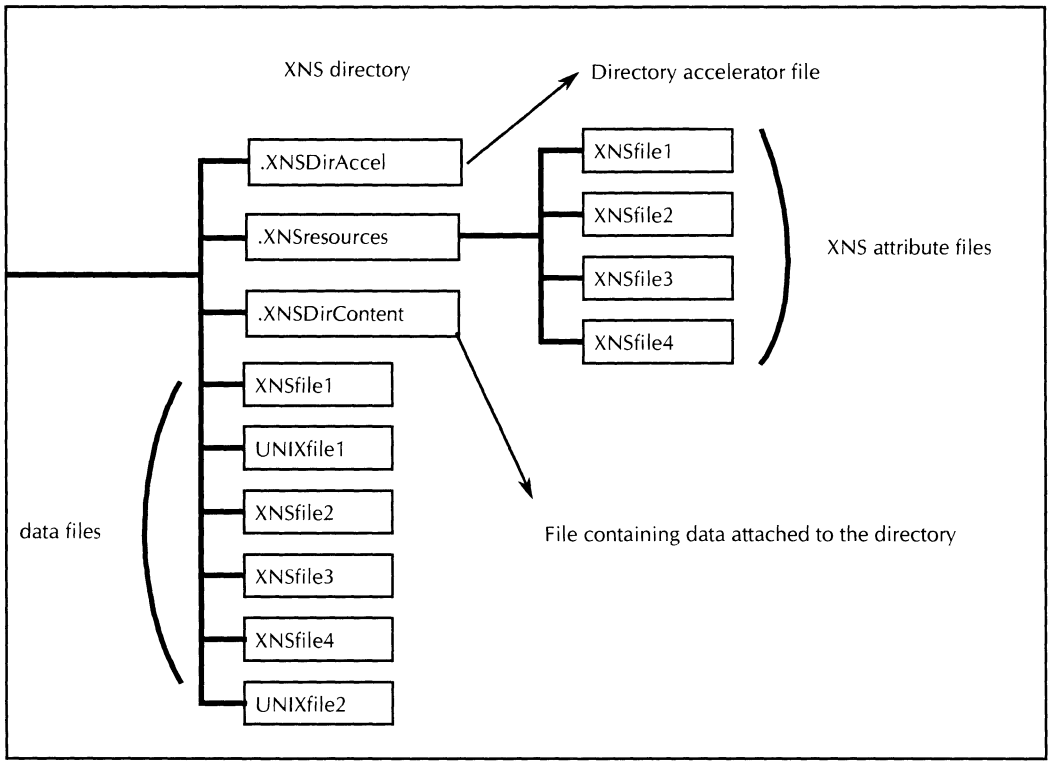
Name

The File Service retains the XNS name attribute within the attribute file. When absent from the attribute file (or if the attribute file does not exist), the File Service truncates the UNIX name to the maximum XNS name length if needed.

File identification

File identifications (FIDs) are the basis of an XNS file reference used by XGV to locate files. XGV rarely uses the name or path of a file to locate that file.

Figure 9-4. **XNS files within UNIX**



The File Service builds the FID from the inode and device number of the corresponding file. The FID is a sequence of five 16-bit words, as follows:

- Random descriptor.
- The middle 32-bits represent the device number (stat.st_dev, refer to the stat(2) UNIX call in the UNIX Reference Manual).
- The last 32-bits represent the inode number (stat.st_ino).

Pathname

The File Service computes the XNS pathname from the XNS name and version numbers of all parent files until the File Service reaches the root of the current XNS volume. The File Service does not escape special characters such as (*),(#),(/),(!) with (').

Type

For UNIX native files, the File Service computes type from the file content. The File Service recognizes the following types:

- 7-bit ASCII files (type 2).
- Interpress files (type 4361).
- GLOBALVIEW RES canvas (type 4428).
- GLOBALVIEW CALS group 4 raster files (type 6501).
- GLOBALVIEW folder for native UNIX directories (type 1).

- GLOBALVIEW file drawer for native directories located within the root of the corresponding XNS volume (type 4498).

The default is the unknown type (type 0).

If the type attribute is present in the attribute file, the File Service computes and overwrites the value.

version

The attribute file retains the version. The UNIX native files version is always 1.

When storing files (using store, create, or deserialize) the File Service uses the client specified version or the next available version number.

datasize

Datasize is maintained by UNIX (stat.st_size).

storedsize

StoredSize is maintained by UNIX (stat.st_blocks) of the context and attribute files

number of children

The File Service computes the numberOfChildren attribute from the number of entries in the UNIX directory.

The File Service does not list or count the following Hidden files:

- .
- ..
- .XNSresources
- .XNSDirContent
- .XNSDirAccel

Ordering

The File Service lists directories in one of the following manners:

- Alphabetically by XNS filename
- By date
- By file position.

File Service sort operations are not case-sensitive.

If the XNS name contains characters which are exclusively in Xerox Character Set 0, the UNIX and XNS name for the file are similar. The main difference is the addition of a version number and possible truncation.

accessList

The File Service retains the accessList within the attribute file. If no accessList is supplied by the client when a file is stored on the file server, accessList is set to "defaulted."

Default access list

Most files use the defaulted access list. In this case, the file drawer determines access to the file.

Glossary

access Set of privileges, each of which, if enabled, allow certain operations to be performed with a container or file drawer. There are five types of privileges: Read, Write, Add, Remove, and Change Access List.

access list List of users or groups of users who have been granted specific types of access to a particular file drawer.

attributes Characteristics of a file such as its name, size, and date of creation. The File Service provides an interface allowing users to set and change six attributes of file drawers (files the File Service maintains): the file drawer name, owner, maximum number of disk pages the file drawer and its contents may occupy, whether its children must be uniquely named, sort order, and access list.

client Software which submits requests to the File Service. All clients interact with the File Service through the XNS Filing protocol. The client determines the data (content) portion of the file. The File Service does not support local filing.

directory File in the hierarchical file system which has other files associated with it and which logically contains those associated files. For example, folders and file drawers are directories.

document File containing text with or without graphics.

file Object in the XNS file system, including documents, file drawers, and file folders. An XNS file consists of an attribute and a data file. Each file has attributes, some of which the File Service maintains, and some of which the client maintains. All files may have content, but no file is required to have content.

file drawer (1) UNIX directory from which clients may store and retrieve files.

(2) Directory in the XNS file server that the File Service defines and maintains, and associates with a user (its owner which can be a user or another service) on the internetwork.

file drawer Temporary identifier the File Service assigns to either a new file or an existing file when a client requests access to that file. The client presents the file handle to the File Service in subsequent operations so that the File Service recognizes the file. A file handle remains valid either until the session ends, or when the client closes the file; it is valid only during the session in which the File Service assigns it to the client.

File Service Service which provides a file storage, and retrieval server to clients on the internetwork. This term can refer either to the named system of files accessible to network clients or to the software which provides a user interface for the definition and maintenance of file drawers within the File Service volumes.

file server Physical hardware on which a File Service runs.

file folder Directory which is defined by the GLOBALVIEW workstation and contains other files (such as documents and other folders) which were created on the workstation. A folder stored on a File Service is contained in a file drawer. A File Service provides no facilities for manipulating the contents of file drawers; a folder may be manipulated only from a workstation.

full access Full privileges for manipulating a file drawer and its contents. Full access to a drawer includes all five types of access (Read, Write, Add, Remove, and Change Access List) and allows the user to perform any operation on the files contained in the file drawer, including changing the access list for a file drawer.

service Software which provides a service to clients of the internetwork. This term refers to the software, not the hardware on which it runs.

server Hardware on which services run. The term refers only to the hardware, not the software running on the machine.

sort order File drawer attribute specifying how the File Service sorts the files in the file drawer. The files may be sorted by name or modifiedOn date, and may be in ascending or descending order. Only a System Administrator can change the sort order.

system administrator User that has special privileges for setting up and maintaining the services. The term can refer either to the user or to the type of privileges a user might have. In each domain of the internetwork there is at least one user that has System Administrator privileges. The System Administrator can perform any File Service operation on any of the file drawers and desktops existing on the File Service volumes. This includes changing and deleting any file drawer or desktop on the File Service.

user Person who operates a Xerox workstartion.

volume File Service directory on a UNIX hard disk partition. The File Service registers volume names in the Clearinghouse Service so that clients can access files within the volume. In the Clearinghouse Service, the volume is registered as a File Service and it is the File Service entity which clients must identify to access a file. Each volume is registered as a separate File Service in the Clearinghouse Service, and the File Service manages all of them.

10. File Service error and information messages

This chapter lists all error messages the File Service may display. Many of these messages display when system administration functions are performed. The remaining messages display at the user's workstation.

If you receive an error message that you do not understand, refer to the alphabetized entries in this appendix for an explaination.

Before calling Xerox service, you should perform the following operations:

1) Retry the operation.
2) Reboot the client.
3) Reboot the server.
4) Reinstall the SCSX and File Service software from media. This will not cause data loss but ensures that no executable files are corrupted.

Indeterminate or insufficient access

At times a user will attempt to access a file drawer and will obtain an indeterminate or insufficient access response. When the response is that the user has indeterminate access, it signifies that the access privileges the user has to the file drawer could not be determined because the CHS is unavailable. This will only occur if the user is not explicitly listed in the access list and if a group is listed in the access list for that drawer.

If the response is that the user has insufficient access, then the File Service has determined that the user does not have access to the drawer, neither as an individual nor as a member of any group which has access to perform the desired operation on the drawer.

If the user is listed in the access list for the drawer, then the indeterminate or insufficient access response should never occur.

Service unavailable or unknown

A user attempting to access a remote File Service may receive the response that the File Service is unavailable. This can occur for a number of reasons, including the following:

- The File Service is in the stopped state.
- The File Service is on a server which is unavailable due to a hardware problem.

- The File Service is on a server which is unavailable due to a software problem.

If the user attempts to access a File Service and receives the response that File Service is unknown, one of the following has occurred:

- The volume is closed.
- The CHS entry for the File Service associated with the volume has an incorrect server address.
- No volume by that name is on the server.

CHS unavailable

The File Service requires that the CHS be available to complete some of its commands and will respond with an error message if the CHS is unavailable. These commands are **Add File Drawer**, **Change File Drawer**, and **Online Volume**.

The reason that the CHS is required to complete the **Add File Drawer** or the **Change File Drawer** command is that the owner of the file drawer must be a valid user. The File Service must access the CHS to determine the validity of the owner name. Further, if attempting to change the access list for a drawer, the File Service must validate each individual or group with the CHS. If the CHS is unavailable, the drawer will not be added or the change to its owner name or access list will not be made.

When a volume is brought online, the File Service must validate that the name of the volume is properly registered in the CHS. If it is known that the CHS entry for the volume being brought online is incorrect (for example, the volume has never been registered, the volume's name has been changed, or the volume has been moved to a new server) then the volume should be taken offline and placed back online when the CHS becomes available.

General error messages

This section contains the error messages which may appear when executing File Service commands. The error messages are listed alphabetically.

Attempting to delete old daemons..Failed.
Please have the Network System Administrator remove p10.v5 on /usr/new/xns/servers.
Please have the Network System Administrator remove p10.v4 on /usr/new/xns/servers.

Explanation When you use the **Install Service** command, the old daemons on /user/new/xns/servers will be deleted if the /usr partition is on your local machine. If your machine mounts the /usr partition remotely, you must be a Network Administrator to delete files on /usr/new/xns/servers. You do not have sufficient access rights to delete files on /usr/new/xns/servers.

Action Have your Network Administrator delete the old daemons on /usr/new/xns/servers.

AUTHENTICATION ERROR: <Description>

Explanation The Clearinghouse Service was unavailable to authenticate your credentials. <Description> may be one of the following:

Inappropriate Primary Credentials
Primary Credentials Expired
Primary Credentials Invalid
Secondary Credentials Required
Secondary Credentials Type Invalid
Secondary Credentials Value Invalid
Verifier Invalid
Verifier Reused

Action

(1) Check your login name and password. Try to log on again.

(2) Check that your server/workstation is connected to the network and that the Clearinghouse Service is running.

Call of 58adm x 1 > tabs/ROOT failed.

Explanation The user interface tool for File Service administration was not invoked because of a UNIX process failure.

Action

(1) Reinstall the File Service software from tape.

(2) If the problem persists, fill out the Problem Report Form and call the Xerox Customer Service Center.

CLEARINGHOUSE ERROR: <description>

Explanation The File Service could not access the Clearinghouse Service. The Clearinghouse Service is too busy or unavailable. <Description> may be one of the following:

Access Rights Insufficient
databaseOverflow
Domain does not exist
illegalDomainName
illegalObjectName
Illegal Property
missing
noChange
noSuchObject
objectOverflow
Organization does not exist
Other
outOfDate
Server Down
Too Busy
unspecifiedError
Use Courier
wrongType

Action

(1) Use the Clearinghouse Service [Show Status] command to verify whether the Clearinghouse Service is started. If it is not, use the Clearinghouse Service [Service Start] command to start it.

(2) Check that your machine is properly connected to the network.

Courier REJECT: <Description>

Explanation A problem occurred when the File Service tried to access the network. <Description> may be one of the following:

Invalid Argument
No Such Procedure Value
No Such Program Number
No Such Version Number

Action Check that your workstation or server is connected to the network and try the operation again.

Desktops cannot be deleted.

Explanation You tried to delete the specified desktop with the **Delete File Drawer** command.

Action Retry using the **Delete Desktop** command.

ERROR: fill_array(): bad malloc()
ERROR: fill_array(): bad realloc()

Explanation A memory allocation problem occurred while processing commands from the Executive.

Action **(1)** Retry the command.

(2) If the error persists, reboot. Try the command again.

ERROR: get_entry(): entry <name> not found.

Explanation You used the **Expunge Service** command in the File Service context. The File Service data has been corrupted as a result.

Action Reinstall the File Service software from tape.

ERROR: get_entry(): entry CallCmd not found.

Explanation The command configuration table has an invalid format, or the File Service cannot find a required entry.

Action All File Service files have been deleted. Exit from the UNIX shell.

ERROR: no valid command table.

Explanation The command configuration tables determine which command is available. One of the command configuration tables may have been deleted.

Action Reinstall the File Service software from tape.

FILING ERROR: <Description>

Explanation An internal File Service problem occurred. <Description> may be one of the following:

Aborted
Access Error
Access Rights Indeterminate
Access Rights Insufficient
Allocation Exceeded
Attribute Area Full
Attribute Type Error
Attribute Value Error
Authenticate Error
Cannot Authenticate
Checksum Incorrect
Connection Error
Control Type Error
Control Value Error
Directory Required
Disallowed
Duplicated
File Changed
File Damaged
File In Use
File Not Found
File Not Unique
File Open
Format Incorrect
Handle Error
Illegal
Insertion Error
Invalid
Loop In Hierarchy
Medium Full
Missing Argument
Missing Courier
Missing Procedure
Missing Program
No Redevous
No Response
No Route
Null Disallowed
Other Call Problem
Parameter Inconsistency
Position Unavailable
Protocol Mismatch
Range Error
Return Timed Out
Scope Type Error
Scope Value Error
Service Already Set
Service Error
Service Full
Service Unavailable
Session Error
Session In Use
Space Error
Token Invalid
Too Many Local Connections

Too Many Remote Connections
Transfer Error
Transmission Hardware
Transport_Timed_Out
Unreasonable
Unimplemented
Wrong Direction

Action **(1)** Retry the command.

(2) If the error persists, reboot. Try the command again.

Illegal character.

Explanation You entered a name using one of these illegal characters:

* [ ] < > # , ; : '

Action Make sure the spelling and format are correct. Refer to the *System Administration Guide* for naming conventions. Enter the correct name and retry the operation.

Illegal name.

Explanation The name was not registered in the specified Clearinghouse Service.

Action Make sure the spelling and format you used are correct. Retry the operation. If the message reappears, use the appropriate Clearinghouse Service [List] command to check the Clearinghouse database for the entry. Retry the operation using a listed name.

Indeterminate access rights.

Explanation The File Service could not determine your access rights to the file drawer. The Clearinghouse Service may be temporarily unavailable or you may be a member of an unregistered group.

Action **(1)** Retry the operation later. If the message reappears, use the Clearinghouse Service [Show Status] command to make sure the Clearinghouse Service is running.

(2) Use the Clearinghouse Service [List Groups] command to make sure your group is registered. If it is, retry the operation. If it is not, use the Clearinghouse Service [Add Group] command to register the group. Then try the operation again.

Insufficient access rights.

Explanation You do not have sufficient access to perform the operation.

Action **(1)** Make sure you are enabled, then retry the operation.

(2) If the message appears again, have a Domain or Organization Administrator use the Clearinghouse Service [Change Domain Access] command to grant you administrative access privileges to the appropriate domain or organization.

Invalid File Service registration.

Explanation The volume is not registered.

Action Retry the operation and enter the service description when prompted.

Invalid UNIX directory.

Explanation The directory you specified as a symbolic link does not exist.

Action Check the UNIX name. Create the directory if necessary. Then retry the operation.

Name not registered.

Explanation When logging on, you specified a user name that was not found in the Clearinghouse Service database.

Action **(1)** Retry the operation using the correct name.

(2) Use the Clearinghouse Service [Service Compare] command to ensure database consistency.

No active volumes.

Explanation There are no volumes on the File Service.

Action Use the **Create Volume** command to create a new volume.

No online volumes.

Explanation The File Service did not find any online volumes.

Action Use the **Online Volume** command to make a volume accessible.

No services are available to install.

Explanation After you expunged the File Service, you tried to reinstall it with the **Install Service** command.

Action Reinstall the File Service software from tape.

REINSTALL or PRODUCT_FACTOR PRODUCT! <Description>

Explanation The execution credentials or file verifiers for the command tables became invalid (unauthorized, copied, or changed). <Description> may be one of the following:

Invalid command file
Invalid command table <table name>
Invalid command table tabs/ROOT

Action Reinstall the File Service software from tape.

Registering of 58adm failed, error code: <number>

Explanation The user interface tool for administrating the File Service was not invoked because of one of the following:

- Bad credentials
- A runtime error, such as a program error, an abnormal program stop, or a system interrupt
- Invalid authorized programs or files.

Action

(1) Reinstall the File Service software from tape.

(2) Reboot the hardware and reinstall the software. If the problem persists, fill out the Problem Report Form and call the Xerox Customer Service Center.

Reset Scavenger Log failed.

Explanation An internal error occurred during the reset scavenger log operation.

Action Try the **Reset Scavenger Log** command again. Reboot the machine. If the problem persists, contact the Xerox Customer Service Center.

Scavenging failed.

Explanation The **Scavenge Volume** command could not repair the volume.

Action Try the **Scavenge Volume** command again. Reboot the machine. If the problem persists, contact the Xerox Customer Service Center.

Server <name> registration failed. <error code> <Description>

Explanation The server could not be registered. <Description> may be one of the following:

Cannot read StableData entry
Execution failed or clearinghouse problem
Invalid function
Program error
Unknown problem

Action

Cannot read StableData entry
The .StableData file has an entry with an invalid format or has been corrupted. Reinstall the File Service software.

Execution failed or clearinghouse problem
A problem occurred communicating with the Clearinghouse Service. Make sure that the Clearinghouse Service is running. If the problem persists, reinstall the File Service software from tape.

Invalid function
The UNIX message catalog for multinationalization could not be accessed. Reinstall the File Service software. If the problem persists, contact the Xerox Customer Service Center.

Program error
Reboot the workstation/server and try the operation again.

Unknown problem
Reinstall the File Service software from tape. If the problem persists, call the Xerox Customer Service Center.

Server <name> unregistering failed. <error code> <Description>

Explanation The server could not be unregistered. <Description> may be one of the following:

Execution failed or clearinghouse problem
Invalid function
Program error
Unknown problem

Action

Execution failed or clearinghouse problem
There was a problem communicating with the Clearinghouse Service. Validate all required entries in the .StableData file. Make sure that the Clearinghouse Service is running. Reinstall the File Service software from tape. If the problem persists, call the Xerox Customer Service Center.

Invalid function
The UNIX message catalog for multinationalization could not be accessed. Reinstall the File Service software from tape. If the problem persists, call the Xerox Customer Service Center.

Program error
Reboot the workstation/server and try the operation again.

Unknown problem
Reinstall the File Service software from tape. If the problem persists, call the Xerox Customer Service Center.

Show Server <server name> failed. <error code>: <Description>

Explanation A problem occurred when you tried to use the **show server** command. <Description> can be:

Execution failed or clearinghouse problem
Program error
Unknown problem

Action

Execution failed or clearinghouse problem
Make sure the Clearinghouse Service is running. Retry the operation. Reinstall the File Service software from tape.

Program error
Reboot the workstation/server and try the operation again.

Unknown problem
Reinstall the File Service software from tape. If the problem persists, fill out the Problem Report Form and call the Xerox Customer Service Center.

System Error.

Explanation An internal File Service problem occurred.

Action Try the operation again. Reboot the server. If the problem persists, fill out the Problem Report Form and contact the Xerox Customer Service Center.

That file drawer is in use. Please try again later.

Explanation The specified file drawer was in use when you tried to access it.

Action Retry the operation later. Use the **Show Activity** command to see who is accessing the volume with the file drawer.

The Clearinghouse cannot be found. The volume cannot be brought online at this time. Try again later.

Explanation There were problems communicating with the Clearinghouse Service.

Action Check that the hardware on whih the File Service is contained is connected to the network. Check to make sure the server/workstation on which the Clearinghouse Service is installed is running. Restart the service, if necessary. Reboot the server/workstation running the File Service. If the problem persists, fill out the Problem Report Form and call the Xerox Customer Service Center.

The delete operation cannot complete. The specified volume is linked to a non-empty directory.

Explanation The **Delete Volume** command did not delete the volume because there are still files or file drawers in the UNIX directory.

Action Use the **List Volumes** and **List File Drawers** commands to check the status of the directory. Use the **Delete File Drawer** command to delete unwanted file drawers.

The desktop directory does not exist.

Explanation The **Delete Desktop** or **List Desktops** command could not complete because the volume does not have a Desktops directory.

Action If you do not need a Desktops directory on this volume, no action is required. Otherwise, move all files and file drawers to another volume. Delete the current volume, and recreate it. A new Desktops directory should be created.

The file drawer name is not unique.

Explanation The name you entered is not acceptable. No changes are made to the existing Clearinghouse database entries.

Action Enter a more specific name and try the operation again.

The File Service cannot be stopped.

Explanation An internal error occurred.

Action Retry the operation. Reboot the server. If the error persists, fill out the Problem Report Form and call the Xerox Customer Service Center.

The file service is not registered in the clearinghouse.

Explanation The volume you specified is not registered.

Action Register the volume, entering the service description when prompted.

The file service net address differs from the local address.

Explanation The volume is not properly registered.

Action Register the volume, entering the service description when prompted.

The file service net address is missing.

Explanation The File Service is not registered or is not properly installed.

Action Use the **Register Service** command or reinstall and initialize the File Service.

The profile file was damaged. Profile has been reinitialized.

Explanation The /etc/xnsfs/config file was incorrect. It was reinitialized to the default values.

Action Use the **Show Profile** command to check the profile. Use the **Change Profile** command to change the profile. You may also need to edit /etc/xnsfs/config to change other parameters.

The specified desktop name was not found.

Explanation The File Service could not find the desktop you specified.

Action Use the **List Desktops** command to make sure the desktop exists on the File Service. Retry the operation using a registered desktop name. Make sure you spell the name correctly, and that it is fully qualified.

The specified file drawer already exists.

Explanation You tried to add a file drawer using the name of an existing file drawer on the File Service volume.

Action Use the **List File Drawers** command to see the names of the existing file drawers. Retry the operation using a unique name.

The specified file drawer was not found.

Explanation The File Service cannot find the file drawer you specified.

Action Use the **List File Drawers** command to list the file drawers on the File Service. Retry the operation using the correct name.

The specified volume is not a file service. Please change the volume name.

Explanation The volume name you specified is already in use.

Action Change the volume name and try again.

The specified volume was not deleted. The volume name is not a file service. Please change the volume name.

Explanation The volume name you specified is already in use by another user.

Action Change the volume name and try the operation again later.

The specified volume was not found.

Explanation You entered an incorrect or invalid volume name.

Action Check the volume name and try the operation again.

The strong key is missing.

Explanation The volume or File Service is not registered properly.

Action **(1)** Register the volume, entering a service description when prompted.

(2) If problems persist, reboot the workstation/server. Try the operation again.

The volume is busy.

Explanation You tried to access a volume already being accessed by another client.

Action Try the operation again later.

unspecifiedError

Explanation An internal File Service problem occurred.

Action Try the operation again. Reboot the server. If the problem persists, fill out the Problem Report Form and contact the Xerox Customer Service Center.

User, alias, or group name could not be validated.

Explanation You entered a Clearinghouse Service user, alias, or group name with an illegal character, improper format, or incorrect pattern.

Action Make sure the spelling and format are correct. Refer to the *System Administration Guide* for naming conventions. Enter the correct name and retry the operation.

Volume could not be brought online. Scavenging may be required.

Explanation The volume was not brought online because of an internal problem.

Action Scavenge the volume, then try the **Online Volume** command again.

Volume link deleted. You must delete files in the linked volume manually.

Explanation The **Delete Volume** command does not always delete all UNIX files in a linked volume.

Action Use UNIX commands to delete other files in the directory.

Warning: Network number in server profile does not agree with net number on the internet.

Explanation When you tried to register the server, you entered a name already in use by another service on another network.

Action Make sure the server name you used is correct. If not, enter the correct name and retry the operation.

11. File Service volume repair

The File Service determines its consistency during each online volume operation. Each Network Service volume maintains an operation log which contains the critical operations for the volume which are currently executing. A critical operation is one that places the volume in an inconsistent state.

When the File Service is about to perform a critical operation, it writes an entry to the operation log, performs the critical operation, and removes the log entry. If there is a File Service failure, when the File Service restarts, it checks the operation log for entries. The entries contain sufficient information for the scavenger to repair the inconsistent volume.

Scavenging

Scavenging generally involves only a few levels of the XNS file hierarchy; however, in some cases the entire volume is scavenged. The only reason for an entire volume scavenge is data loss during a UNIX system call, a lost operation log, or a database error caused by corrupt or lost File Identification table entries.

The scavenger is invoked automatically when volumes are brought online and the volume is inconsistent. The system administrator can scavenge a volume manually using the **Scavenge Volume** command. Manual scavenge is used when you discover volume damage that was undetected by the File Service.

Scavenger corrective action

Corrective action performed during the scavenge consists of the following:

- The scavenger moves files that are in the wrong directory to the correct location. (During an NSFile move operation, the content and attribute files may be in separate directories for a short time.) The scavenger deletes the results of partial copy, or deserialization operations. The client does not lose data since copy and deserialization procedures do not affect the source files.
- The scavenger deletes damaged attribute files. Attribute files are corrupt if they have an invalid checksum or if they have an incorrect file length. In either case, the scavenger reads the existing attribute file until it detects an error. When an attribute file is deleted, the content file continues to exist and is interpreted as a UNIX file. These files have a version.

- The scavenger recalculates the subtree sizes for each directory, based on the UNIX file size, and writes this to each attribute file.

Scavenger causes

The following are NSFile operations that can cause a scavenge and the corrective actions that the scavenger takes.

- Deserialize – If a file or subtree of files is not completely deserialized to the new location, the entire subtree of files is removed.
- Delete – If context and attribute files exist, no action is taken, otherwise, a system failure occurs after the content file is removed. The scavenger then removes the attribute file and updates the subtree size of the attribute files.
- Copy – If a file or subtree of files is not completely copied to the new location, the entire subtree of files is removed.
- Move – If the content file is moved but the attribute file exists in the source directory, copy the content file back to the source directory. If content and attribute files are in the destination directory, verify the attribute file's contents and update the subtree sizes of the source and destination directories.
- Change Attribute – Verify the attribute file.

The scavenger deletes content files when the source for the files still exists. If the data in an attribute file is unreadable, the scavenger deletes the attribute file.

12. NFS restrictions

The File Service supports remote XNS volumes through the Sun Network File System (NFS) protocol. NFS offers customers many advantages such as storing or accessing XNS and native files on different types of hardware.

Security is a concern. The File Service uses NFS as a communication protocol between the server and remotely stored files, the security of the file service is limited by the security of NFS. (In a similar fashion, the security of locally stored XNS files is limited by the security of UNIX.)

Connectivity

NFS allows an almost infinite combination of networked hardware. Since it is impossible to test this large number of hardware configurations, the only NFS configurations that the File Service officially supports are the following:

- NFS mounting is only supported at the volume level, as Figure 12-1 illustrates. Mounting an NFS directory or file drawer is not supported or documented.
- Only soft mounts are supported to avoid hanging the File Service when an NFS server is unavailable.
- The SunOS partition that contains the File Server needs sufficient room for structures supporting the NFS mounted volume. These structures consume approximately 128 bytes for each XNS file stored. (The exact amount of space required depends on the file name length and how deeply the stored files are nested.) An NFS mounted volume containing 1000 files would require about 128 kilobytes of storage for the FID table. Other data structures are stored in the remote NFS directory or are insignificant in size.
- Only machines that run SunOS Version 2 NFS are supported.

These configurations allow two basic types of access: gateway, and multi-protocol server. The gateway access allows XNS volumes to be mounted remotely on other hardware, as Figure 12-2 illustrates. The multi-protocol server configuration allows NFS access to XNS volumes located on a SunOS server; this approach is suited to high capacity servers. There are no restrictions to prevent combining both approaches.

The multi-protocol server approach can possibly be extended to include portable Netware, LAN Manager X, TOPS for Mach, and so forth. Refer to your Sun documentation to verify configurations.

Figure 12-1. **NFS**

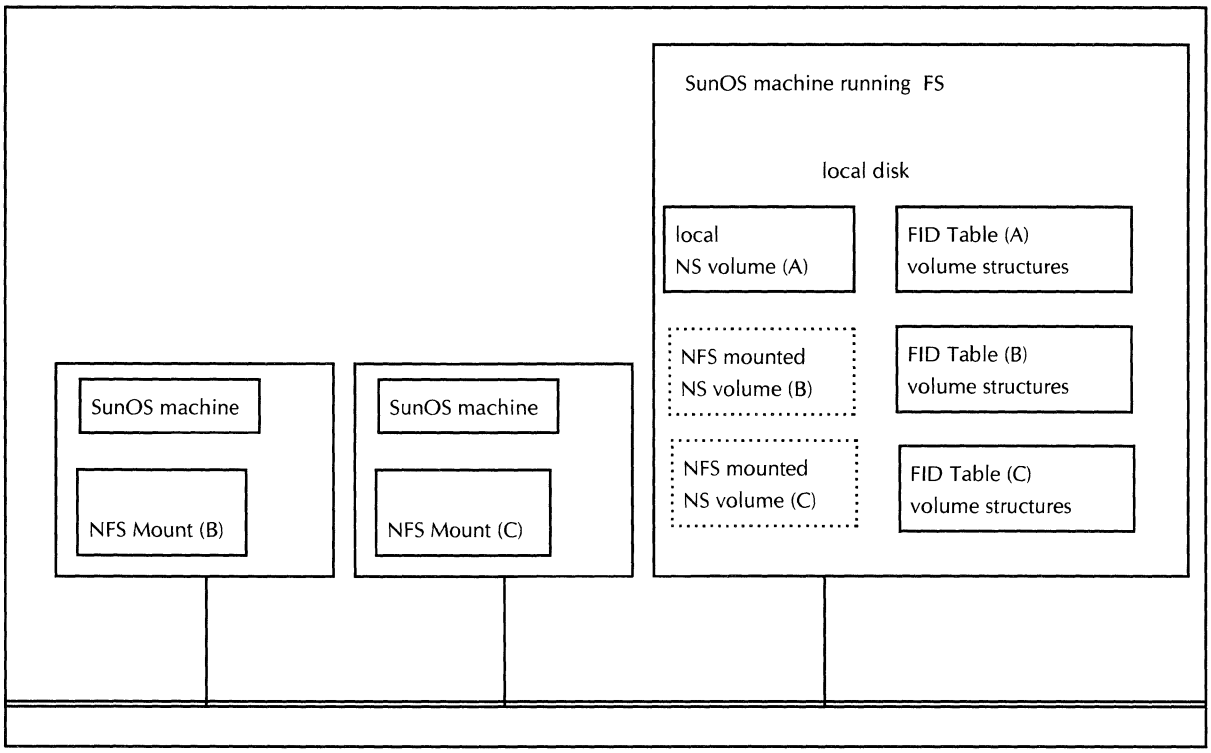
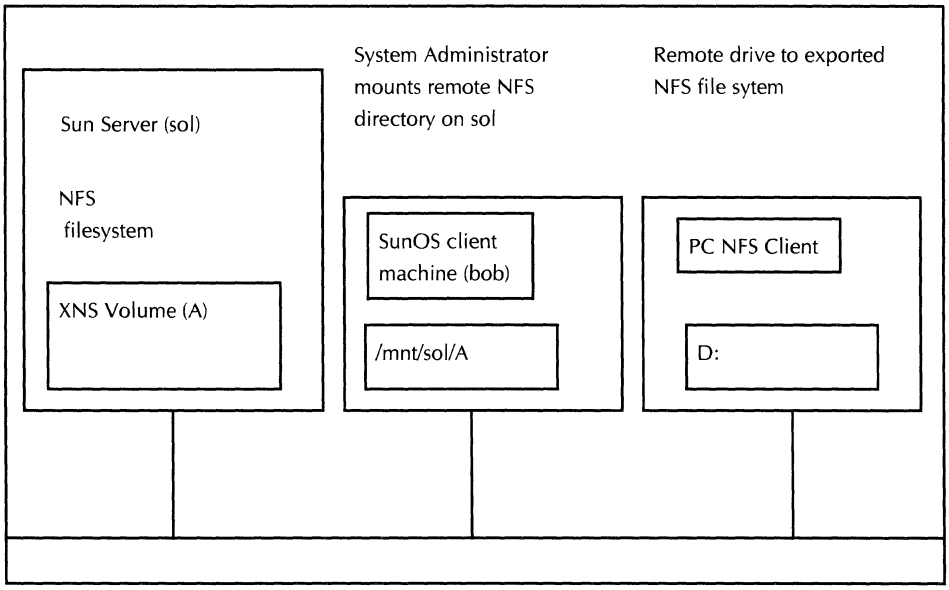


Figure 12-2. **Remote links**



Network File System security

NFS version 2 provides authentication of users through the Sun remote procedure call (RPC); UNIX Authentication is provided by DES encryption for credentials. (See Remote Procedure Calls: Protocol Specification). The level of security NFS version 2 offers is much higher than in the past. The main security issues, due to mounting NFS volumes, are as follows:

- The File Service requires root access to the remote SunOS hardware. The File Service changes the owner of each stored file in both the remote and local cases. This UNIX chown requires root privileges. If a customer will not allow root access to the NFS mounted machines, then the File Service will not support remotely mounted volumes.

  NOTE: The File Service will store the files on the NFS server, but when the File Service attempts to chown the file, a UNIX error occurs. Thus, the File Service may leave unuseable files on the NFS server.

- If the NFS server responds but refuses to allow access to the specified file, the File Server returns the error, "InsufficientAccess," to the client.

- NFS (and UNIX in general) does not have a global networked authentication service. Since there is no central authority, it is impossible to ensure verification of user access.

Network File System specific problems

An NFS server may fail or become unavailable for a number of reasons. If the volume containing the NFS mount is unavailable, the File Server returns the error, "FileNotFound" to the client. Local NS volumes remain available in this event.

If an NFS filesystem is mounted with a hard mount and the NFS service is not responding, the File Service will hang. The File Service requires soft mounts to avoid this hang.

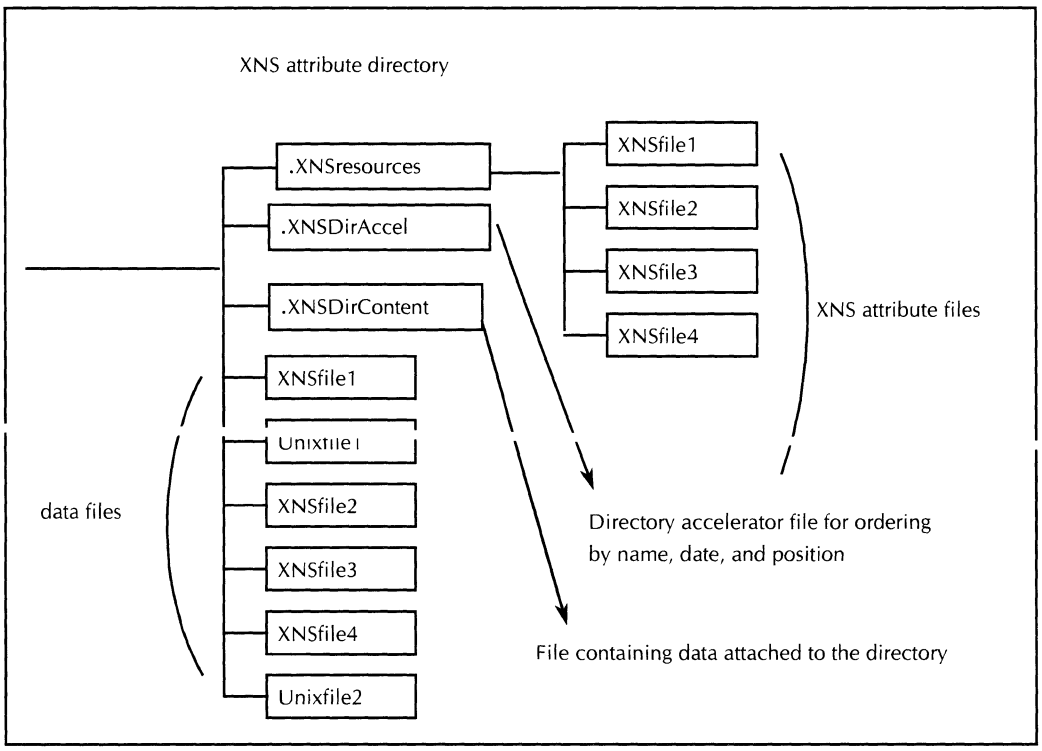
If the NFS server responds but refuses to allow access to the specified file, the File Server returns the error, "InsufficientAccess" to the client.

Many different vendors support NFS on their UNIX operating systems. In most respects, the File Service can handle these different versions of UNIX; however, the maximum filename length and other attributes may be different on different systems. The File Service is not tested for these operating system dependencies.

13. File Service backup

The File Service does not provide a specialized backup and restore facility. Customers should use native UNIX utilities (such as dump, tar, bar) or third party utilities to back up XNS files. Since XNS files are composed of two UNIX files (content and attribute), both files must be restored to the proper locations (refer to Figure 13-1).

Figure 13-1. **XNS files within UNIX**



Restoring files

The following guidelines must be followed when restoring XNS files from UNIX backups:

- Determine the XNS filename to restore
- If the XNS filename does not use Roman characters, determine the corresponding UNIX filename using a Xerox provided translation tool.

  For English names, the UNIX filename at each level will be the same as the XNS name. The File Service appends an extension (consisting of a period, tilde, version number, tilde; .~01~) to the UNIX file.
- Restore the specified file (for example, backup/.../foo.~01~) to its original location; this is the content file.
- Restore the file (e.g backup/.../.XNSresources/foo.~01~) to its original location; this is the attribute file.
- If the file is a directory, restore all of its descendants.

The most common mistake when restoring XNS files is forgetting to restore the attribute file.

UNIX System Administrators should not back up an XNS volume when the volume is online; otherwise, there may be inconsistencies in the backed-up files.

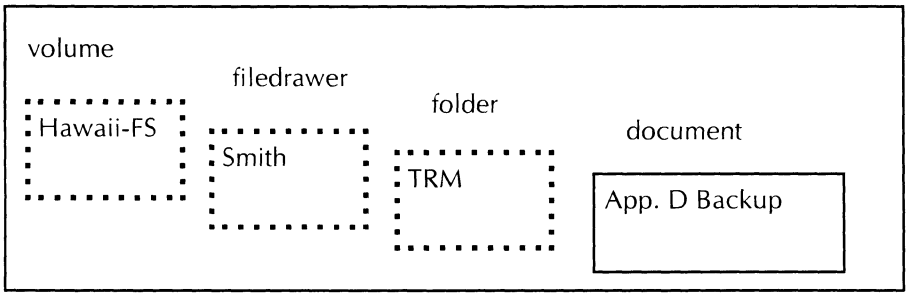
System Administrators should not restore the volume FID tables for the following reasons:

- The FID tables for each XNS volume are sparse UNIX files (containing holes). If a System Administrator backs up the FID table for a volume, these holes are filled in, and the FID table takes up a large amount of space on the backup media.
- Restoring a backed-up FID table is only valid if the backup process returns the NS File volume to the exact state in which the FID table is in (for example, If the disk is lost and ALL files stored on the volume are restored). If single files are restored, the values stored in the FID table for the restored files are not valid.

Sample file restore

This example contains the informatiopn necessary to restore the XNS file with the pathname /Hawaii-FS/Smith/TRM, as illustrated in Figure 13-1. :

Figure 13-1. **Sample file path**



The corresponding UNIX content pathname you use to restore the XNS file "Backup" is /XNSfileServers/Hawaii-FS/Smith.~**01**~/**TRM.~01~**/App. D Backup.~**01**~

The corresponding UNIX content pathname you use to restore the XNS file "Backup" is /XNSfileServers/Hawaii-FS/Smith.~**01**~/**TRM.~01~/.XNSresources**/App. D Backup.~**01**~

The following are descriptions of the UNIX pathname structure:

- /XNSfileServers – is the root of all XNS volumes
- Hawaii-FS – corresponds to the XNS volume Hawaii-FS. Volume names do not have extensions.
- Smith.~**01**~ – corresponds to the XNS file drawer Smith. File drawer names have extensions that are usually set to ~01~
- TRM.~01~ – corresponds to an XNS folder, book, or other container named TRM. If this was version 5 of the TRM, the extension would be .~05~.
- App. D Backup.~01~ – is the UNIX filename that corresponds to the XNS file "Backup". The UNIX name may contain spaces that must be escaped or placed in quotes when accessing the file using UNIX.

14. UNIX/XNS access

This chapter explains how the File Service manages access to files, and how UNIX access permissions are set for stored files.

Introduction

The File Service stores the XNS access rights to a file in a corresponding XNS attribute file (for example, shared books maintain an access list). However, access rights are generally defaulted in these attribute files and access rights are determined at the file drawer level.

When determining access, the File Service recursively searches the ancestors of the file for a nondefaulted access list. If the File Service encounters a UNIX file during this search, the File Service uses the UNIX access rights of the content file to determine access.

The server determines access rights, as follows:

- Uses the explicit XNS access rights of the specified file.
- Searches for XNS access rights in ancestor files. This usually locates the rights at the file drawer level.
- If a non-XNS file is found as an ancestor, before an access list is found, the File Service uses the UNIX rights.

At sites where UNIX users can add files to the XNS hierarchy, the XNS users may encounter access restrictions. The XNS user expects access to all files stored in the file drawer they own, as the XNS access at the file drawer level shows. However, if a UNIX user stores a file in a directory within this file drawer, the owner of the file drawer cannot access that file unless the owner has matching UNIX access rights.

The following sections discuss XNS access lists, UNIX file ownership, how to determine file access, and how to extend the interoperability between XNS and UNIX.

XNS access lists

Normally, XNS access to a file is determined by a stored XNS access list. The access list contains the names and permissions allowed for a user, group, or wildcard pattern. The File Service sets the access list for each file drawer. The following are examples of Clearinghouse Service names, groups, and wildcard patterns that may appear in an access list

- Jane Smith:South:ABC Co.
- Services Development:North:DakotaCo
- *:*:*
- *:North:Xerox
- *:*:Xerox

The type of access determines what operations a user can perform on the file drawer, and on the files within that file drawer. The operations that a user may perform when granted each type of access are, as follows:

- Read – allows Read access to the files within a file drawer. A user with Read access may list the contents of, and copy, objects contained in the file drawer.
- Add – allows Add access to the file drawer. A user may add an object to the file drawer. (Other access permissions are not required to add an object.)
- Remove – allows the user to remove an object from the file drawer if the user also has Write access to the file drawer. In general, the user should have Read access to the drawer to allow the current contents of the file drawer to be displayted before performing a delete operation.
- Write – A user with Write access may change the attributes of objects in the file drawer (such as the name). In general, the user should have read access to the file drawer to allow the current contents of the file drawer to be displayed before changing the attributes of the contents.
- Change Access List – allows a user to change the access list privileges of the file drawer.

A user may have any combination of these access rights. An access list may contain any number of entries.

When you create a file drawer for a user, that user becomes the owner of that file drawer, and is given full access (all five types of access rights). The owner of a file drawer need not be on the access list for the file drawer.

Any user having Change Access List privileges to a file drawer may change the access list for that file drawer. You may change the access list of any file drawer.

XNS and UNIX file ownership

When the File Service stores a file with the XNS protocol, the file has a set of XNS access rights (usually inherited from its parent) and a set of UNIX access rights. There are separate UNIX access rights for the content and attribute files which together make up a single XNS file.

The XNS access rights are the same manner as for the Xerox 8090 File Service.

The UNIX owner of the attribute file is always set to root. Access to the attribute file is restricted to all other UNIX users.

The UNIX owner of the content file has its owner set according to the UNIX name in the XNSusers file. This ASCII XNSusers file contains XNS Clearinghouse Service names and the equivalent UNIX names.

XNS users file structure

Each XNSusers file entry has the following format:

ENTRY TYPE#XNS NAME#UNIX User Name#Comment ..

This release does not use the "ENTRY TYPE" field.

The "XNS NAME" field retains the XNS user name. This must be an XNS user distinguished name (cannot be alias or group). Because the File Service comparison is not case sensitive, capitalization is not important in this field. The File Service allows only ASCII characters in the XNS name, and does not allow Clearinghouse Service names that have accents, or other special characters.

The "XNS NAME" field supports limited wildcards, as follows:

- *:*:* – addresses all XNS users.
- *:*:XF – represents all the users of the XF organization.
- *:CB19:XF – represents all names in the CB19 domain.
- Fred:CB19:XF – represents the user Fred in the CB19 domain.

The "UNIX User Name" field contains the UNIX user name which is equivalent to the "XNS NAME" field. This name should exist in the file "/etc/passwd" (refer to the UNIX user names and passwords) or the NIS.

The "Comment" field may contain any comment. The order in which entries are stored in this file is not important.

The following is a sample XNSusers mapping file:

```
#*:*:*#nobody
#*:*:Xerox#xfcomm
#*:North:Xerox#Bill
#JOHN:North:Xerox#John#John Smith
```

The File Service determines the UNIX access rights to a file by searching the XNSusers table for the most restrictive name that matches the XNS user storing the file. (*:North:Xerox is more restrictive than *:*:Xerox.)

For example, if the XNS user "John:North:Xerox" stores a file, the UNIX owner of the content file is set to John. (JOHN:North:Xerox exactly matches an entry in the XNSusers file.)

If the XNS user "Jane:North:Xerox" stores a file, the UNIX owner of the content file is set to Bill (Jane:North:Xerox matches the first three entries, but the most restictive match is *:North:Xerox.)

Access determination

If there are no UNIX files stored in the XNS hierarchy, file access is determined by the XNS access rights. However, if UNIX files exist, users may experience unexpected access behavior. Table 14-1 lists each possible access scenario, and how the File Service will determine file access.

Table 14-1. **Access scenarios to UNIX and XNS files**

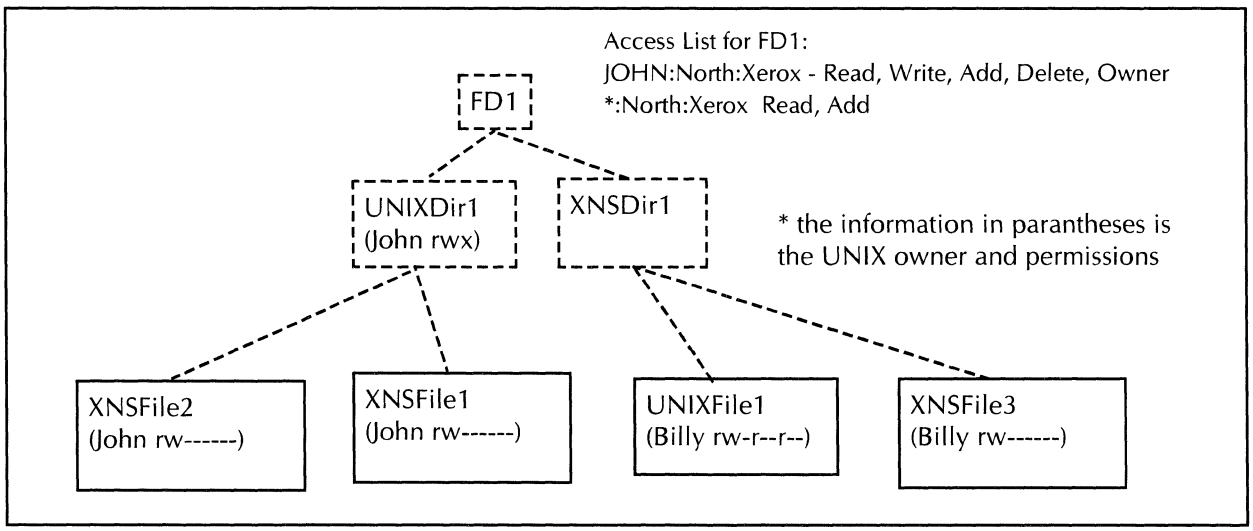
Parent	File	Access
UNIX	UNIX	UNIX access of file
UNIX	XNS	UNIX access of parent (unless the XNS file has an explicit access list)
XNS	UNIX	UNIX access of file
XNS	XNS	XNS access of parent (usually determined at the file drawer level)

If UNIX access is used, the File Service will determine the UNIX equivalent access for the XNS user by using the XNSusers file. The following examples should make access determination more clear.

The following examples use an XNSusers file containing the following data and files structured as shown in Figure 14-1.

```
#*:*:*#nobody
#*:*:Xerox#xfcomm
#*:North:Xerox#Bill
#JOHN:North:Xerox#John#John       Smith
```

Figure 14-1. Access information for a sample file drawer



Example 1 Unexpected access denial - The owner of the file drawer, (JOHN:North:Xerox) attempts to delete the file "UNIXFile1." The deletion fails since the access to UNIXFile1 is determined by the

UNIX access. The File Service will map JOHN:North:Xerox to the UNIX user John, and John has only Read access to UNIXFile1. This access restriction may suprise JOHN:North:Xerox since the XNS access list on FD1 gives him deletion privileges.

Example 2 Storing XNS files under a UNIX directory - JOHN:North:Xerox stores the file XNSFile1 under UNIXDir1; XNSFile1 will not be readable by other users due to the UNIX access restrictions on UNIXDir1. Only XNS users that map to the UNIX user John are able to read XNSFile1 and XNSFile2. This behavior is unexpected since the access list for FD1 gives read access to *:North:Xerox.

Example 3 Setting the UNIX owner of a stored XNS file - JOHN:North:Xerox stores the file XNSFile2 under UNIXDir1. The File Service will determine from the XNSusers file that the UNIX user John maps to JOHN:North:Xerox; the owner of XNSFile2 is thus set to John.

If Fred:North:Xerox stores XNSFile3 under XNSDir1, the File Service will determine, from the FD1 access list, that Fred:North:Xerox has Add access. When the File Service creates the content file, it checks the XNSusers file for the most restrictive name (*:North:Xerox) that matches Fred:North:Xerox. The content file is then created with the UNIX equivalent user of *:North:Xerox, which is the UNIX user Bill.

Modifying XNS user files

You should be very careful when modifying the XNSusers file to preserve the security of XNS files. Some types of interoperability require you to change the XNSusers file.

The default XNSusers file to customers has only the single entry:
::*#nobody

You can add entries to the XNSusers file to allow one-to-one or many-to-one access.

One-to-one XNS/UNIX access

One-to-one access is useful when both XNS and UNIX files are contained on a File Service and UNIX users want to access files using their XNS accounts.

By modifying the XNSusers file, the File Service can grant access to a specific UNIX user. Such a modification allows for the sharing of files between one or more XNS users and a certified UNIX user.

For example, if you add#Jane Smith:North:Xerox#JSmith to the XNSusers file, you allow access only to the UNIX user named JSmith. This scheme is useful if JSmith and Jane Smith:North:Xerox" are accounts representing the same person. The JSmith user can access her XNS files from UNIX, but no other UNIX users can access files stored by "Jane Smith:North:Xerox." Likewise, the JSmith user cannot access files that another UNIX user creates and stores.

Many-to-one XNS/UNIX

Many-to-one access is useful in a large XNS community that requires a small set of UNIX users or daemons to access XNS files. For example, a single person may be responsible for merging XNS report files into a UNIX database.

Similarly, files that a UNIX program stores may need to allow a set of XNS users access. An example of this might be a third party application that generates bitmaps that it incorporates into a Xerox GLOBALVIEW document.

UNIX access by a single UNIX account to all files stored by a set of XNS users requires the following entry:

```
# *:West Valley:Xerox#Sjohnson
```

This allows the UNIX user, Sjohnson, to access a file that any user stores in the "West Valley" domain. The Sjohnson job might entail retrieving XNS files and converting them for use in UNIX. Access control with this type of modification is very strong since the File Service allows only a few select UNIX users access to XNS files.

The UNIX user SJohnson can store files only to directories in the "West Valley" domain. Again, this allows a highly restricted Add or Remove access to the XNS world from a UNIX account.

An XNS user file can have multiple entries of one-to-one or many-to-one.

Modifying the configuration file

The File Service configuration file contains the UNIX access mode that the File Service uses when storing the file using XNS filing. This mode determines the amount of access that other UNIX users have to XNS files. By default, access is given only to the UNIX equivalent owner of the file. If you require more file sharing between XNS and UNIX, you may reset this value accordingly.

The /etc/xnsfs/config file contains the UNIX protection mode field for content files (refer to the chmod command in the SunOS Reference Manual). The default of this field is 700 (Read, Write, and Execute permission) ensuring that only the UNIX equivalent owner of the XNS file has any access from UNIX. Changing this value allows group or world access of other UNIX accounts.

The UNIX chmod command uses an octal code to change UNIX file permissions. With the File Service, you can change file permissions using the same format as the chmod command.

To update the /etc/xnsfs/config file, you must be an enabled System Administrator. Use the File Service **Change Profile** command, and enter the new mode at the UNIX File Mode prompt. File protection and permissions in UNIX are Read, Write, and Execute (rwx). Changing the octal code of this line changes the file permissions as follows:

- Protection_mask 700 – only owner has access.

- Protection_mask 740 – owner has access, and group has Read access.

- Protection_mask 744 – owner has access, group has Read access, and world has Read access.

A protection_mask setting of 744 allows UNIX users to read the XNS files stored by XNS users. This setting is useful when you want to give Read access to XNS files from UNIX accounts. UNIX users cannot modify the XNS file data they can only read it.

If an XNS file drawer does not allow Read access to all XNS users, then granting UNIX access is an XNS security violation. The acceptability of this violation must be determined on a site-by-site basis. This scheme is most appropriate for servers that contain public file drawers and is not appropriate for a private file drawer.

Allowing any UNIX user to write to XNS file drawer hierarchies is allowed when the the mask is set to one of the following:

- Protection_mask 720
- Protection_mask 722.

This configuration has the greatest potential for misuse and data corruption. Any UNIX user will be able to write new files and delete existing XNS files or components of XNS files. A UNIX user may also modify an XNS content or attribute file and cause a server failure.

Allowing Write protection for a UNIX group is more acceptable but still allows a fairly large number of individuals to modify the data in an XNS volume.

Allowing UNIX users to have Execute access to XNS files does not make sense in most cases since data is stored in serialized format.

15. File Service executable files and data

This chapter shows the File Service component file, locations, and functions. The location of each component assumes that you have installed the File Service from some media and have executed the Service Common Software for UNIX Install Services command.

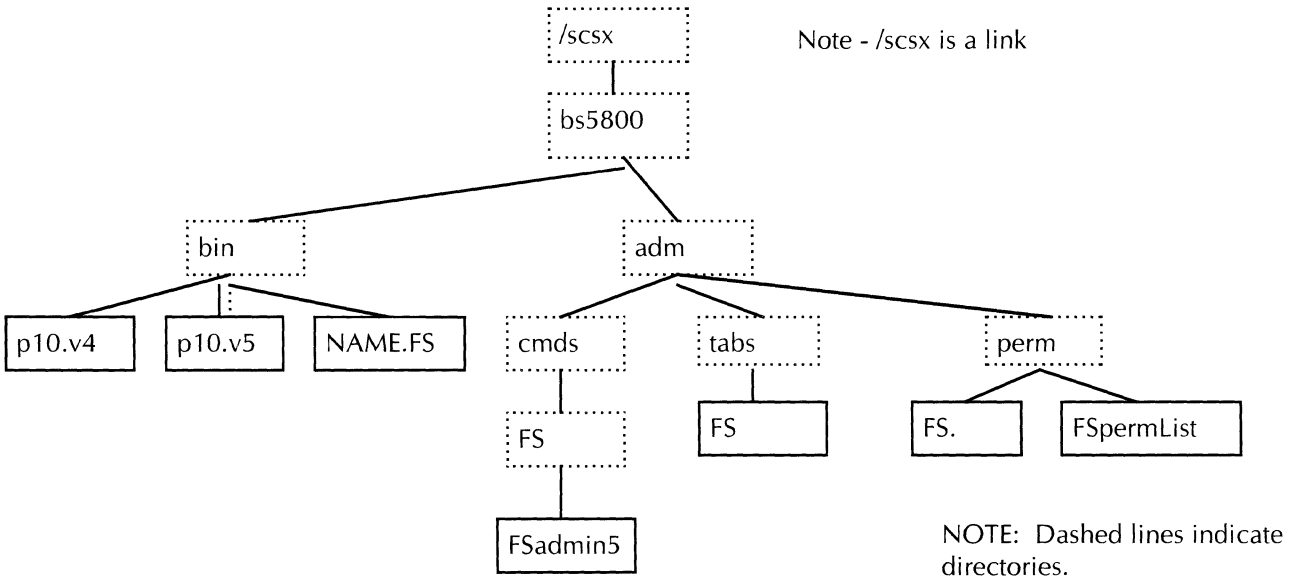
Introduction

During installation, the /etc/passwd and /etc/xnsup files are modified. A new login account is added to /etc/passwd; this account is used for system administration. The xnsup file is modified to notify that the XNS helper software of the location of the File Service executable code.

Most of the remaining File Service files are stored in the UNIX path /scsx/bs5800. The following files are stored in the /etc directory:

- /etc/XNSusers
- /etc/xnsfs/config
- /etc/rc.xnsfileservice.

Figure 15-1. Main File Service executable files and data files



The remainder of this appendix describes each File Service file and lists the problems that can occur if the file is damaged.

Files stored in the /etc directory

The File Service installation updates or stores several files outside the main /scsx/... file tree. Files are stored outside /scsx if there is a possibility that a user may edit them.

- The /etc/passwd file is a standard UNIX file and the File Service installation only appends to it.
- The /etc/xnsup file is owned by the lower level XNS software and is modified by the installation scripts.
- The /etc/XNSusers file is stored in an easily accessible location since it may be modified by administrators
- The /etc/xnsfs/config file is rarely modified directly by System Administrators.
- The /etc/rc.xnsfileservice file should never be modified; it executes during the hardware reboot.

/etc/passwd

The installation scripts append the following line to the /etc/passwd file:

xnsadm::580:580:FS System Administrator:/scsx/bs5800:bin/csh

When you perform a UNIX login with the xnsadm account, an SCSX program 58adm is started. 58adm manages the administrative session.

The UNIX System Administrator may edit the /etc/passwd file and change the xnsadm entry. If the /scsx/bs5800 field is changed or missing, then the .login file for the xnsadm account cannot be found. If SCSX does not run after xnsadm login, then the /etc/passwd file may have caused the problem.

/etc/xnsup

The installation scripts modify the /etc/xnsup file to include the location of the File Service executables. The /etc/xnsup file should include the following line:

/etc/xnshelper $* "default domain" "North" "default organization" "Xerox" "courier server dir0" /scsx/bs5800/bin "courier server dir1" /usr/new/xns/servers &

The values "OSBU North", and "Xerox" will be your domain and organization name respectively. The important parts of this command line are:

"courier server dir0" /scsx/bs5800/bin

This line tells the XNS software that it should look for a service program in the /scsx/bs5800/bin directory. (Refer to Figure 15-1 for the File Service executable programs p10.v5 and p10.v4).

If /etc/xnsup is missing the /scsx/bs5800/bin field, then clients will not be able to connect to the File Service. The message, "No such service" or "No such program number" is sent back to the client software.

All XNS service programs written in C use the naming convention "p<program-number>.v<version-number>"

The program number for the File Service is 10 and versions 4 and 5 are supported.

/etc/XNSusers

The /etc/XNSusers file is used to assign UNIX access rights to XNS files. By default, all XNS files have the UNIX owner set to "nobody." The XNS files are thus protected from UNIX access since no user can login to the "nobody" account.

By adding entries to /etc/XNSusers, you can allow UNIX access to XNS files. See the "UNIX and XNS Access" chapter for more information on how to modify the /etc/XNSusers file.

/etc/xnsfs/config

The File Service stores its configuration data in the /etc/xnsfs/config file. The most commonly used fields in the config file can be manipulated using the File Service administrative session. Other fields in this file can be modified by Xerox personnel via special scripts. If the config file is improperly edited from UNIX, it may become corrupt. A corrupted configuration file is ignored and the File Service uses hard-coded default values.

The following shows the names of the File Service configuration data files:

xns_access_list
debug_messages
debug_output
debug_attributes
protection_mask
debug_locks
debug_file_locks
bdt_fork
ns_volume_location
max_contexts
max_children
max_handles

The rest of this section shows the structure, content, and default values of the preceding files.

xns_access_list — This configuration file should always be set to y (yes).

debug_messages — Displays several messages on the console for each filing request received when this item is set to y (yes). This item is only for on-site debugging.

debug_output — Determines what UNIX device receives the debug messages. This value should be set to /dev/console.

debug_attributes — Displays type and value of attributes exchanged between the File Service and client. The debug_attributes operates only if "debug_messages" is set to y (yes).

protection_mask

Specifies the mode in which files are created when stored on UNIX. This item should be a three digit number in octal format (see chmod(1) in the UNIX Reference Manual). The default mode is Read, Write, and Execute (rwx) for the owner (700) and no permission for other users. The mode value greatly determines the security of XNS files that are on the File Service. Refer to the "UNIX and XNS Access" chapter for details on how to modify this value to allow interoperability between native UNIX files and XNS files.

debug_locks

Displays messages regarding context slots and allocation table locking in the context shared memory area. The debug_locks operate only if "debug_message" is set to y, and is for debugging purposes.

debug_file_locks

Displays messages regarding file locking at the XNS level, locks requested by filing clients, and implicit locks obtained by the File Service itself. The debug_file_locks operates only if "debug_messages" is set to y (yes).

bdt_fork

Forks a child process for handling bulk data transfers between file server and clients when this option is set to y (yes). The bdt_fork should always be set to y. The bdt_fork is for debugging purposes.

ns_volume_location

Specifies the location for all XNS volumes for this file server. This value is always set to /XNSfileServers. During installation, a link (/XNSfileServers) is created that points to the disk partition containing the XNS volumes.

The ns_volume_location (/XNSfileServers) is a UNIX link to the directory that contains then File Service volumes. If File Service volumes suddenly become unavailable ("serviceUnavailable"), you should check to make sure that /XNSfileServers exists and that it points to the proper directory. The File Service will display an error message in the console if /XNSfileServers does not exist.

max_contexts

Specifies the maximum number of NSFile sessions for the server. The default is 32 sessions (you set this value during File Service installation, and you should not increase it arbitrarily since each session consumes one semaphore).

The default value for max_contexts (maximum File Service sessions) is 32 in the configuration file. If the configuration file is damaged or missing, the File Service will reduce this to 20 sessions as a minimal configuration.

max_children

This value is not used.

max_handles

Specifies the maximum number of open files allowed for a specified NSFile session. The max_handles item may need to be increased beyond 58 if a client requires a very large number of open files. The network installer requires that this value be set to 130.

The default value for max_handles is 58 in the config file. If the configuration file is damaged or missing, the File Service will reduce this value to 32. The

max_handles item must be at least two times greater than the maximum depth of your XNS file structure. For example, if max_handles is 58, the File Service can support a maximum XNS file structure depth of about 29 levels. Since there is some additional handle overhead, the File Service can support only 25 sessions with 58 handles.

If the config file is missing or damaged, max_handles is defaulted to 32. The File Service can support about 13 or 14 levels of nesting with max_handles set to 32. If a user receives an NSFile Undefined error 768 while operating on a deeply nested file, max_handles may be too small. If the nested depth is 14, the configuration file may be damaged.

Some File Service clients require that max_handles be increased. If the File Service is used for an installation drawer, you may need to increase max_handles to 130.

/etc/rc.xnsfileservice

The /etc/rc.xnsfileservice is a daemon scavenger that ensures that the File Service daemons are in a valid state. The rc.xnsfileservice file is invoked by the rc.local file each time the hardware is rebooted.

rc.xnsfileservice renames +p10.v4 and +p10.v5 to _p10.v4 and _p10.v5 respectively. The plus sign (+) indicates that the File Service daemons were in the stopping state when a hardware failure or reboot occurred.

If rc.xnsfileservice is missing or damaged, the daemons may remain in the stopping state. If this should occur, simply rename the /scsx/bs5800/bin/+p10.v4 and /scsx/bs5800/bin/+p10.v5 manually and reboot the server.

Files stored in the /scsx/bs5800 directory

The /scsx/bs5800 contains File Service and SCSX files that should not be modified. Most files stored under /scsx/bs5800 are protected by security mechanisms that invalidate modified files. If you accidentially edit a protected file, you must reinstall SCSX and the File Service from media.

The following sections describe each file stored in the /scsx/bs5800 directory that is owned by the File Service. The remaining files are part of SCSX and are described in a separate appendix.

/scsx/bs5800/bin

The /scsx/bs5800/bin directory contains the File Service daemons p10.v4 and p10.v5. If the daemon names are preceded by an underscore (_), the File Service is in a stopped state. If the daemon names are preceded by a plus sign (+) the File Service is in the stopping state. If the daemon names are not preceded by a character, the File Service is started.

If for any reason the File Service state is inconsistent with what the administration session expects, you should do either of the following:

- Reboot the hardware (normal action).
- Change the daemon names to _p10.v4 and _p10.v5 (stopped state) and start the File Service from the administration session. If manually changing the name does not work, reboot the hardware.

The /scsx/bs5800/bin directory contains another File Service file (NAME.FS) that is used by the administration session to determine what services are on the machine. NAME.File Service contains the long (File Service) and short (FS) names for the File Service.

If NAME.FS is missing, the administrative session will not know that there is a File Service installed.

NAME.FS is a protected file; if it is modified in any way, you must reinstall Service Common Software for UNIX and the File Service from media.

/scsx/bs5800/cmds/File Service/File Serviceadmin5

The fsadmin5 file is a protected file which contains the File Service administration commands, and can be invoked only through the protected SCSX user interface. If fsadmin5 is missing, you will receive a message in the SCSX administrative session that looks like the following:

```
/scsx/bs5800/tmp/5800.1254  cmds/FS/fsadmin5: not found
```

If the fsadmin5 file has been modified, you will receive the following message when executing a File Service administrative command:

```
***** REINSTALL or PRODUCT_FACTOR PRODUCT Invalid
command file
```

You must reinstall the File Service from media if the fsadmin5 file is deleted or modified.

/scsx/bs5800/adm/tabs/FS

File Service is a state table that is used by SCSX to determine which File Service commands to make available. File Service is a protected file, and if it is modified in any way, you must reinstall the File Service from media. You will see the following message if the File Service file has been modified.

***** REINSTALL or PRODUCT_FACTOR PRODUCT Invalid command table: tabs/File Service

/scsx/bs5800/adm/perm/FS

The File Service. file is available to store product factoring and version information. This file is not used in this software version. If this file is removed or modified, there is no impact on the File Service or SCSX.

/scsx/bs5800/adm/perm/FSpermList

The FSpermList file is a protected file that contains the pathnames of all File Service administrative components. If FSpermList is modifed, you must reinstall the File Service from media. You will receive the following message if this file is modified.

***** REINSTALL or PRODUCT_FACTOR PRODUCT Invalid command table: tabs/File Service

This message also displays if the tabs/File Service file is modified.

/var/adm/xnsfslog

The xnsfslog file contains all messages that the File Service sends to the console window. These messages include the following:

- File Service running out of various resources
- Missing /XNSfileServers link or other critical files
- System or File Service internal errors.

The xnsfslog file is never truncated; thus, it may become quite large. If your root partition is running out of room, you may want to delete the xnsfslog file to free up some space.

Index

Index