Site Operations

Introduction to Site Operations and Administration

This document provides information for Site Administrators, people who set up ("configure") and maintain sites with one or more Symbolics computers. Typically, the Site Administrator is responsible for the following tasks:

- Installing new releases of Genera software, and any layered products used at the site.

- Choosing a network addressing scheme. Each computer needs a unique network address. All Symbolics computers run Chaosnet, and each Symbolics computer needs a Chaosnet address. Some sites also use IP/TCP or DECnet protocols; if so, the computers there need Internet or DECnet addresses as well as Chaosnet addresses.

- Choosing a configuration for the site. In a site with more than one computer, you need to decide how to distribute responsibility among them. Most sites designate one Symbolics machine as a server; this machine might be a namespace server, file server, and mail server. Users of the other computers store their files on the server, which means you need only do backups on one machine, instead of on several. If you have a printer, you need to decide which computer will be the print server. Large sites might designate several machines to be servers.

- Configuring the software to reflect the site itself. The namespace database describes the site and the computers, networks, printers, and users in it. You need to keep the namespace database current with the configuration of your site.

- Customizing worlds for various purposes. Sometimes it is useful for a site to have different worlds running on Symbolics machines for different purposes. For example, a server machine requires a world with the server software in it. Other machines require worlds with site-specific customizations and applications.

- Backing up files to tape. As in other computing environments, it is important to establish a regular schedule for copying the contents of disks to tape.

- Serving as a liason with Symbolics Customer Service (for example, sending "crash data" mail to Symbolics at CUSTOMER-REPORTS@STONY-BROOK.SCRC.SYMBOLICS.COM or CUSTOMER-REPORTS@SCRC.STONY-BROOK.ARPA).

Note: Genera software installation procedures and site configuration information are documented in the Software Installation Guide for each new release of the Genera software. For software installation information about layered products, see the documentation that is shipped with each layered product.

This document contains reference information about tools and techniques for Site Administrators. Because this is a reference document (and because each site is different), you will not need to master all of the information provided here. Only portions of this document will be relevant to you, and to your site's needs.

In particular, this document includes the following information:

Booting the Symbolics Machine
: When you boot the Lisp machine, you bring up a "world". The world is your Lisp environment. A Lisp world can reside either on the local disk of the machine you wish to boot, or on a remote machine (called a net-boot server). This section describes cold booting, warm booting, netbooting, and autobooting.

Setting Up and Maintaining the Namespace Database
: The Site Administrator sets up and maintains the namespace database by registering new hosts, printers, and users, and otherwise updating the namespace database so that it "knows about" any changes to your site's configuration. This section describes how to edit the namespace database, and the semantics of the information stored in it.

Making, Distributing, and Using Worlds
: When a new release of Genera is available, the Site Administrator needs to make new worlds for the site to use. The new world must contain both the new Genera software and information about the site itself. This section describes how to make new worlds, and how to distribute them among computers at the site.

The Front-End Processor
: When you boot a Symbolics computer, you communicate with the Front-End Processor (the FEP). This section describes how to use the FEP commands, and gives other information about the FEP.

The Lisp Machine File System (LMFS)
: All Symbolics computers use Lisp Machine File System (LMFS) files and directories. This section provides information about utilizing your disk to maximize the space available for LMFS files (by using LMFS partitions). This section also describes the procedures for backing up, dumping, reloading, and retrieving files (and systems) in LMFS.

Using the Tape Facilities
: Symbolics supports different tape formats for different purposes. This section describes how to know which format to use for a given purpose, and how to use the tools for writing the contents of a disk to magnetic tape.

Symbolics Store-and-Forward Mailer
: The Mailer is a program that provides mail forwarding and delivery services to users at a site. The Mailer is distinct from Zmail, which is a user program for reading and composing mail. The Mailer program runs on a Symbolics machine designated as a *mail server*. This section describes how the Mailer works, and how to perform administrative tasks such as setting up mailing lists.

Internet Domain Names
: The Internet Domain Names system is a collection of specifications and procedures which implement the DOMAIN protocol, which is commonly used on the ARPA Internet. The DOMAIN protocol deals extensively with naming, and it distributes the responsibility of maintaining naming information across a set of administrative bodies. This section describes the Symbolics implementation of the DOMAIN protocol.

Symbolics Dialnet
: Symbolics Dialnet supports the international dial network, meaning that it enables communication over the telephone lines. Dialnet is primarily used for mail transfer.

Note that several other documents contain important related information. We particularly recommend the following sections:

- For an introduction to the anatomy of a Symbolics computer, including the FEP and other parts of the machine, see the section "Workbook: Overview of the Machine".

- For introductory information on networks, using server machines, and how the namespace database describes the network configuration, see the section "Concepts of Symbolics Networks".

- For reference information on network addressing, see the section "Network Addressing".

- For information on choosing a network addressing scheme, see the section "Choosing a Network Addressing Scheme".

Booting the Symbolics Machine

You give commands to the FEP in order to boot the Symbolics machine. When you boot, you bring up a "world". The world is your Lisp environment. A Lisp world can reside either on the local disk of the machine you wish to boot, or on a remote machine (called a netboot server). FEP commands to boot a world can be entered manually (at the `FEP Command:` prompt), or executed from within a file (using the Boot FEP command).

Boot File Types

Boot files always have the extension `.boot`. Three types of boot files exist. Each boot file type has an initial default filename (provided by the FEP):

`boot.boot`
: This type of file contains commands for booting a specific Lisp world. It's useful to have several boot files, especially if your site uses various world loads, some of which may have special programs loaded into their environment.

Boot files can incorporate either the FEP Netboot command (to netboot from a remote host) or the FEP command Load World (to boot a world from the local disk).

Note: Always press RETURN at the end of a boot file before saving it.

`hello.boot` — This type of file contains commands for both scanning FEP overlay (flod) files, and initializing the local machine's hardware tables. More information is available about the FEP overlay (flod) files.

- See the section "Overlay (Flod) Files and the FEP".
- See the section "Scanning the Overlay (Flod) Files".

`autoboot.boot` — This type of file combines the functions of `boot.boot` and `hello.boot` files. It contains commands for scanning the FEP overlay (flod) files, for initializing the hardware tables on the local machine, and for booting a specific Lisp world. See the section "Autobooting".

Contents of Hello.Boot Files and Boot.Boot Files

This section briefly describes the philosophy of which FEP commands belong in the `Hello.boot` file and which belong in the `Boot.boot` file on both 3600-family and Ivory-based machines.

Hello.boot File

This file should contain anything that holds constant for this machine, regardless of the world you are running. For example, this includes your flod files, your paging files, your network address, and so on.

First, this file must scan all the flods from which you want to use commands. This includes the lisp, loaders, info, and debug flods.

Next, the file should include the Initialize Hardware Tables command, so that any errors it encounters will be seen early in the booting process. (This command is completely optional in Ivory-based machines. It is automatically invoked by the Ivory FEP when needed, but errors it signals may be less clear when automatically invoked.)

Then, the `Hello.boot` file should declare any paging files you always want to use; this saves you having to manually Add Paging Files, or to say the same in your boot file. Do not declare any "spare" paging you want to save for an emergency, because Load World automatically uses all declared paging files, and thus the spare paging file would not be available in case of an emergency. It is harmless to declare paging files that do not exist (for example, one you might delete every now and then to make room for Copy World); nonexistent ones give a warning, but are not used.

Ivory machines only:

Starting with Genera 8.1, the `hello.boot` file for an Ivory machine should contain two lines:

```
Hello Innn
Hello Local (or hostname)
```

Each of these corresponds to a boot file. The *nnn* refers to the IFEP number, which is 325 for Genera 8.1.

Hello I*nnn* Boot File

The `I`*nnn*`.boot` file (where *nnn* is the IFep version number, which is 325 for Genera 8.1) should contain the commands to scan the flod files and initialize things.

```
Scan I325-lisp.flod
Scan I325-loaders.flod
Scan I325-info.flod
Scan I325-debug.flod
Initialize Hardware Tables
```

Hello Local Boot File

The `local.boot` file should contain those commands that set up this specific machine, declaring paging files, setting the network address, and any other boot options.

```
Declare Paging Files FEP0:>Paging-1.page
Declare More Paging Files FEP0:>Paging-2.page,Paging-3.page
Set Boot Options :Network Address Chaos|52525 :IDS Enable
```

Boot options are required for standalone sites. They are optional for other sites, but they do save manually invoking them or having them duplicated or inconsistent in boot files (if you have multiple boot files). See the section "Set Boot Options FEP Command".

The boot options replace other FEP commands. For compatibility, some of the boot options have corresponding commands. Note that some of these commands (such as Enable/Disable IDS on 3600-family machines), if present, must appear in the `Boot.boot` file, since they must follow Load World. (In the Ivory-based FEP, these commands warn if you try to use them at the wrong time, while in the 3600-family FEP they silently perform no operation if used at the wrong time). We recommend using the Set Boot Options command instead of the corresponding commands, so you can set all the boot options in one spot.

Normally, you set your network address with Set Boot Options, but you could use the Set Network Address command instead. Whichever way you choose to set the network address, it should be done in the `Local.boot` file, and should not be duplicated in your `Boot.boot` files.

3600 and Ivory Machines

Note that, in previous releases, you had to set the network address after loading the world, but this restriction no longer holds on either architecture.

Here is a sample `Hello.boot` file for a 3600-family machine:

Sample 3600-Family Machine Hello.Boot File

```
Scan v127-info.flod
Scan V127-loaders.flod
Scan v127-lisp.flod
Scan v127-debug.flod
Initialize Hardware Tables
Declare Paging-Files FEP0:>Paging-1.page
Declare More Paging-Files FEP0:>Paging-2.page,Paging-3.page
Set Chaos-Address 52525
```

(For a sample `Hello.boot` file for an Ivory-based machine, see the section "Sample Ivory-Based Machine Hello.Boot File".)

Boot.boot Files

These files should contain any setting that pertains only to the particular world (and microcode, on a 3600-family machine) this file loads and starts. (Note that this excludes the address of the machine.) There is a slim possibility that you might want to have some explicit paging file commands here (for example, a boot file to boot with just one paging file explicitly added for when you want to do world copying).

Sample 3600-Family Boot.Boot Files

Here is the sequence of commands for a Symbolics 3600-family machine that is cold booting a world from the local disk:

```
Clear Machine
Load Microcode FEP0:>3640-fpa-mic.mic.430
Load World FEP0:>genera-8-1.load
Enable IDS
Start
```

Here is the sequence of commands for a Symbolics 3600-family machine that is cold booting a world from a remote disk (netbooting):

```
Clear Machine
Load Microcode FEP0:>3640-fpa-mic.mic.430
Netboot inc-site-genera-8-1
Enable IDS
Start
```

The information in the rest of this section applies to Ivory-based machines only.

`Boot.boot` files for Ivory-based machines should contain only the following:

```
Load World
Start
```

Note that you can set your default world to boot in boot options. If you choose to load the default world, there is no need to give an explicit pathname argument to Load World.

It is unnecessary to do a Clear Machine before the Load World since this command does nothing on Ivory-based machines. On 3600 family machines, it serves a purpose. To avoid confusion and make boot files easier to understand and maintain, if Clear Machine is left in the boot file of an Ivory machine, it is ignored.

It is possible, although not recommended, to give the Enable/Disable IDS command here, but the recommended way is to do all that in your `Hello.boot` file, via Set Boot Options :IDS [Enable/Disable]. You should also use the Set Network Address commands in your `Hello.boot` file. (Note that if someone copies a boot file with an address in it to another machine, confusion will result.)

Note also that Disable IDS saves only an inconsequential amount of memory and overhead, so there is usually no advantage in doing it.

Here is the recommended sequence of commands for a Symbolics Ivory-based machine that is cold booting a world from the local disk:

```
Load World
Start
```

This loads the most recent world on your local disk, which is usually the one you want to boot.

For a netbooted machine, the recommended sequence is:

```
Netboot inc-site-genera-8-1
Start
```

Cold Booting

Cold booting completely resets Lisp and puts the machine into a "fresh state" for the next user. You can cold boot a world that is resident on the local disk. Alternatively, you can cold boot a world from remote machine (that is, netboot).

To cold boot a world from a remote host (netboot server), include the FEP command Netboot in your machine's boot file (or in a manually entered sequence of FEP commands). For more information about the Netboot command, see the section "Netboot FEP Command".

If you are cold booting a world from the local disk, include the FEP Load World command in your machine's boot file (or in a manually entered sequence of FEP commands). For more information about the Load World command, see the section "Load World FEP Command".

If the calendar clock has not been set, the machine will notify you, so that you can set it. Use the Command Processor (CP) Set Time command. For more information, see the section "Set Time Command".

Here is the sequence of commands for a Symbolics 3600-family machine that is cold booting a world from the local disk:

```
Clear Machine
Load Microcode FEP0:>3640-fpa-mic.mic.430
Load World FEP0:>genera-8-1.load
Enable IDS
Start
```

Here is the sequence of commands for a Symbolics 3600-family machine that is cold booting a world from a remote disk (netbooting):

```
Clear Machine
Load Microcode FEP0:>3640-fpa-mic.mic.430
Netboot inc-site-genera-8-1
Enable IDS
Start
```

Here is the recommended sequence of commands for a Symbolics Ivory-based machine that is cold booting a world from the local disk:

```
Load World
Start
```

This loads the most recent world on your local disk, which is usually the one you want to boot.

For a netbooted machine, the recommended sequence is:

```
Netboot inc-site-genera-8-1
Start
```

More information is available about boot files. See the section "Boot File Types".

Netbooting

Note: Only Symbolics 3600-family machines that run Genera 7.2 or a later release can be netboot servers. Only Symbolics Ivory machines that run Genera 8.1 or a later release can be netboot servers. A netboot server must be on the same subnet as the user machine it is netbooting.

Netbooting allows Symbolics machine users to boot and run worlds from remote machines. In order to netboot, sufficient paging space (around 40,000 blocks, or space equivalent to the size of the world you are netbooting) must be available on the local disk.

In order to use netbooting, each user also needs a netboot core on the local disk. For more information about netboot cores, see the section "Netboot Cores".

If the same worlds are used by multiple machines at your site, netbooting will enable you to keep — and maintain — only two copies of each world (one of them a backup), rather than one copy of each world for every user.

If you netboot an Incremental Disk Save (IDS) world, all its parents will be netbooted as well. In order to netboot them, IDS worlds and their parents must reside on the same (netboot server's) disk.

Note: The Load World FEP command checks the local disks for an IDS world's parents. If one or more parents is missing, the Load World FEP command will look for the parent on all enabled netboot servers, and attempt to netboot it. This means that, if all the parents of an IDS world do not reside on the local disk, Load World becomes a request to netboot the parent worlds of an IDS loaded from the local disk.

More information is available about IDS worlds. See the section "Using the Incremental Disk Save (IDS) Facility".

If you want to use netbooting at your site:

- Create one or more netboot servers with world-load files for all worlds used at the site. For information about creating netboot servers, see the section "Setting Up Servers for Netbooting".

- Put a netboot core on each user machine that will use netbooting. For information about setting up user machines for netbooting, see the section "Setting Up User Machines for Netbooting".

- Include a Netboot FEP command where the Load World FEP command would otherwise appear in boot files, or in a manually entered sequence of FEP commands. For information about the Netboot FEP command, see the section "Netboot FEP Command".

The FEP command Netboot takes the argument *world-description*.

world-description can be a complete specification, or a short, unique substring from the name of a world-load file. *world-description* need not include a disk unit specification, or the file type `.load` or `.ilod`.

If you have two worlds with identical substrings in them, the Netboot FEP command looks for the newest one. For example, if these two worlds exist:

Genera-8-0-incremental-1.load

Genera-8-0-incremental-2.load

and you issue this command:

```
Command: Netboot Genera-8-0
```

you get the newest world containing the substring Genera 8.0 (but you are not able to specify which one you want to boot).

To ensure that you netboot the world you intend, name the worlds something like this:

Inc-1

Inc-2

Then, issue the Netboot FEP command, followed by the appropriate (unique) sub-string.

Netboot Cores

Netboot cores are small Lisp programs that contain the necessary code to bring a Lisp world across the network and load it into local memory.

You can have more than one netboot core on a disk. It is best to use a netboot core built from the same version of Genera as the world you are booting.

On machines with loadable microcode, the FEP looks for the appropriate netboot core for the world you are booting. The appropriate netboot core is the newest one that will run with the microcode that you have loaded.

Netboot cores contain all of the wired pages for a world (around 150 pages). If you use a netboot core built from the world you are booting, it will use itself as the wired pages for the world and load only the remainder of the world (this saves a small amount of time).

Netboot cores have the file type `.load`, and use approximately 150 blocks each. If you have a Genera 8.0 world-load file on your FEP, you can use this world-load file as a netboot core. Any Genera 8.0 world load can be used as a netboot core, but netboot core files are much smaller.

Create Netboot Core Command

Create Netboot Core *world*

Creates a netboot core, a small world with just enough information in it to be able to locate and boot a world over the net. An appropriate netboot core is included on every distribution tape, so you should never need to use this command.

world {*pathname*} The pathname of a world load file to use to create the netboot core. This world must be resident on your FEP or IFEP.

Setting Up Servers for Netbooting

Note: Only Symbolics 3600-family machines that run Genera 7.2 or a later release can be netboot servers. Only Symbolics Ivory machines that run Genera 8.1 or a later release can be netboot servers. A netboot server must be on the same subnet as the user machine it is netbooting.

To set up a netboot server, perform the following steps:

1. Use the Copy Flod Files command to copy the new overlays (flod files) to the FEP file system on the server running the Genera 8.0 world.

2. Add netboot service to the server's Host Object in the local namespace:

```
Service: NETBOOT SLAP NETBOOT
 . . .
Server Machine: Yes
```

   (`Server Machine: Yes` causes the server machine to boot with services disabled. This prevents users from requesting netboot service before the server itself has finished booting.)

3. Copy any worlds that users will be netbooting. Put them in a top-level FEP directory on the netboot server.

4. Enable services once the netboot server is finished booting. Use the Enable Services CP command to enable all services (including netboot service) like this:

```
Enable Services (disabled services [default All]) All
```

   Alternatively, use the Command Processor (CP) Enable Services command to enable netboot service like this:

```
Enable Services (disabled services [default All]) Netboot
```

Netbooting includes a queuing mechanism; netboot service is provided serially, on a first-come-first-served basis. You can have a backup netboot server. This is useful if, at your site, many machines could require netboot service at the same time (following a power failure, for example).

A user machine can be a netboot server, but this may slow the netbooting process to some extent. A user machine is better utilized as a backup (rather than a primary) server. Symbolics does not recommend using a file server as a netboot server, since file service can degrade during netbooting operations, and vice versa. It's preferable to dedicate one machine at your site as the netboot server.

If you have booted identical worlds on different netboot servers, remember to maintain them both at the same patch level.

Setting Up User Machines for Netbooting

Note: Only Symbolics 3600-family machines that run Genera 7.2 or a later release can be netboot servers. Only Symbolics Ivory machines that run Genera 8.1 or a later release can be netboot servers. A netboot server must be on the same subnet as the user machine it is netbooting.

Set up a user machine for netbooting like this:

1. Use the Copy Flod Files command to copy the new overlays (flod files) to the FEP file system. See the section "Copy Flod Files Command". Copy the netboot core to the FEP file system of the local machine using the Copy World command. See the section "Copy World Command".

This file has the same name as the distribution world, prefixed by Netboot-core-from-, such as Netboot-Core-from-8-0.load. If you have multiple disks, put the core on FEP0.

(If you explicitly mount another disk in your boot file before issuing the Netboot command, you can safely put the netboot core on that disk.)

2. Add extra paging space, equivalent to the size of the world you will netboot, in addition to the paging space you normally use. This extra paging space is necessary if you want the netbooted world to have as much virtual memory space as a locally booted world would.

Place the FEP command Netboot in your machine's boot file where you would normally use the Load World command. For information about the Netboot FEP command, see the section "Netboot FEP Command". Here is a sample boot file with a recommended sequence of commands for using Netboot:

```
Clear Machine
Load Microcode FEP0:>3640-fpa-mic.mic.430
Netboot inc-site-genera-8-0
Enable IDS
Start
```

See the section "Contents of Hello.Boot Files and Boot.Boot Files".

User machines that have been netbooted do not need any world files in their FEP file system except the netboot core (and a backup copy of the core).

If you have a Genera 8.0 world-load file on your FEP, you can use the world-load file as a netboot core. (Any Genera 8.0 world load can be used as a netboot core, but netboot core files are much smaller.)

When you netboot, the screen blanks and then provides a narrative of the netbooting process. You can monitor netbooting via the progress bar at the lower right-hand portion of the screen. The leading, dotted line indicates how much of the world has been requested for loading. The trailing, solid line indicates how much of the world has been loaded.

The label beneath the progress bar describes each phase of the netboot process. Additionally, the short bars indicate (from left to right, respectively) Disk-Wait, CPU-Run, and Net-Wait status.

You can halt netbooting at any time, as indicated in the upper left of the screen. For 3600 family, XL family, and UX family machines you press h-c-FUNCTION. On the NXP1000, press the NMI button on the front of the machine.

If, for any reason, your netboot core does not work properly, you may need to use the FEP command Set World-to-Netboot. For information about the Set World-to-Netboot FEP command, see the section "Set World to Netboot FEP Command".

Autobooting

Ivory-based and 3600-family machines with G208 FEP EPROMS support autobooting. (To find out the FEP version of your 3600-family machine, type the Command Processor (CP) command Show Machine Configuration in a Lisp Listener.)

When a machine is set up for autobooting, it boots automatically at power up. The autoboot command sequence is specified by an `autoboot.boot` file; the presence of the `autoboot.boot` file enables the autoboot process.

The `autoboot.boot` file must include all of the FEP commands that are normally specified by both the `hello.boot` and `boot.boot` files, in the order in which these commands would normally be executed.

Specifically, `autoboot.boot` must contain commands to:

- Scan the required overlay (flod) files.
- Declare paging files.
- Set the Chaos address.
- Initialize the hardware tables.
- Clear the machine.
- Load the appropriate microcode and world.
- Start the machine.

The file `autoboot.boot` must reside on the lowest numbered disk unit implemented in your machine. (In most cases, this is Unit 0.)

Note that the Ivory FEP allows nested boot files, so your autoboot file could contain only the following:

```
Hello
Autoboot Delay 10 (Press a character to stop autobooting now)
Boot
```

During disk drive spin-up, the autoboot software waits for Unit 0 to respond. If, after three minutes, Unit 0 has not responded, the system checks the lowest numbered disk unit for the existence of the `autoboot.boot` file.

On Ivory-based machines, you can abort the autoboot process at any time (by pressing any character). On 3600-family machines, you can abort autobooting — within the first 10 seconds of the process — by pressing any key, or while the Autoboot Delay FEP command is executing. (You can use the Autoboot Delay FEP command to lengthen the time period during which you can abort autobooting.) Once you've aborted autobooting, boot by using the standard `hello.boot` and `boot.boot` files (or power-cycle the machine to begin the autoboot process again).

Additional information is available about the Autoboot Delay FEP command. See the section "Autoboot Delay FEP Command".

Booting IDS Worlds

It's possible to keep an Incremental Disk Save (IDS) world without being required to keep all of its parents) on a local disk. When you want to boot the IDS world, use the Load World FEP command.

Note: The Load World FEP command checks the local disks for an IDS world's parents. If one or more parents is missing, the Load World FEP command will look for the parent on all enabled netboot servers, and attempt to netboot it. This means that, if all the parents of an IDS world do not reside on the local disk, Load World becomes a request to netboot the parent worlds of an IDS loaded from the local disk.

If no enabled netboot server exists, the Start command will start Lisp and Lisp will wait for a netboot server, without returning an error. If your site does not support netbooting, use `h-c-FUNCTION` to halt the netboot process, get back to the FEP, and boot from the local disk.

Warm Booting

Warm booting is a recovery procedure that may enable you to restart Lisp in order to save buffers and mail. It destroys the state of the process running at the time of the boot, destroys the state of the window system, and resets all network connections. If you warm boot after a crash, do not expect to continue running for very long after a warm boot, unless the cause of the crash can be rectified.

Once you've warm booted, save your work, try to determine the cause of the crash, and cold boot the machine. For information about investigating problems that cause your machine to crash, see the section "Debugging in the FEP".

Warm boot the machine by using one of the following procedures:

1. If the machine crashed, issue the Show Status FEP command at the FEP prompt, check the information it provides, and then issue the Start FEP command. For information about checking the information that the Show Status FEP command provides, see the section "Debugging in the FEP".

2. If the machine did not crash, but is unresponsive, issue the Command Processor (CP) Halt Machine command, or press `h-c-FUNCTION` if you cannot get a Lisp Listener or if the Lisp Listener is not responding to keyboard input. (Note: on UX-family machines `h-c-FUNCTION` only works from the cold load stream, so you might have to first enter the cold load stream with `FUNCTION SUSPEND` or by opening the cold load icon in order to enter the FEP.) Then, issue the Start FEP command at the FEP prompt. For information about halting Lisp, see the section "Halting". Alternatively, on MacIvory machines, you can choose Restart Lisp from the Ivory menu item to warm boot Lisp.

Note: On MacIvory machines, choosing Shutdown from the Ivory menu, subsequently choosing Quit from the File menu, and then choosing Restart Lisp Ivory menu item also causes a warm boot.

In this case, though, because you have properly shut down Lisp (instead of having crashed), you can expect to operate normally; your machine's state will be as it was before quitting Lisp (rather than initialized, as it would be after a cold boot).

Setting Up and Maintaining the Namespace Database

The site administrator or site maintainer sets up and maintains the namespace database by (for example) registering new hosts and users, and otherwise updating the namespace database so that it "knows about" any changes to your site's configuration.

Site configuration enables your Symbolics computers to describe and access the resources available to them. Once your site has been configured, all of its Symbolics computers can find out about the other hosts, printers, and users there.

Namespace service is at the heart of site configuration. In order for one machine (a local host) to use any of the resources provided by other machines (remote hosts), namespace service — managed and provided by a namespace server — is required.

A local host depends on the namespace server for answers to these questions:

- How is the remote host connected to the local host (what is the remote host's network address)?
- What network protocol must be used to obtain the desired service?

By configuring your site, you give each Symbolics machine sufficient information to know where and how to obtain namespace service. You also give your namespace server sufficient information to provide that service.

Register new users in the namespace database either before they use the system, or when they log in for the first time. New hosts and printers should be registered in the namespace database before being connected to the network or to the host that will support them.

For more information about the namespace database and why it's used, see the document *Genera User's Guide*.

Site Configuration and Namespace Service

The namespace server uses namespace files to describe each resource at a particular site. The namespace server might not store the namespace files locally, but it knows where to locate them.

If your site is large, with many (over ten) user machines to make demands on the namespace server, Symbolics recommends that you create a dedicated namespace server; use one Symbolics machine that's unavailable for user applications. If your site is small (under ten user machines), designate one of the user machines as the namespace server.

The namespace server's purpose is to collect and maintain information for a site. All of the information known about a site's network(s) and each host, printer, and user is stored in the namespace database. More information is available about namespace objects and the namespace system. See the section "Setting Up and Maintaining the Namespace Database".

There can be more than one namespace server at a site. One server is the primary namespace server; the others are secondary namespace servers. More information is available about the differences between server machines. See the section "Machines and Worlds".

A typical Symbolics site uses a namespace server to store the namespace database, and a file server to store system sources and online documentation. It is possible for one machine to perform both these services, provided it has enough disk space.

There are some restrictions pertaining to servers. The namespace server must be a Symbolics computer. The system sources and online documentation must reside on one of the following:

- A Symbolics computer.
- Any UNIX host running NFS, in a `.sct` directory (see the section "Using SCT with a UNIX File System").

More information is available about system sources and online documentation. See the section "System Sources and Online Documentation".

Using the Set Site and Define Site Commands

If you have a Symbolics 3600-family machine, and you boot a distribution world (the base world that Symbolics sends you), the Command Processor (CP) Show Herald command displays the machine's name as *DIS-LOCAL-HOST* and the site name as *DISTRIBUTION*.

- In order to use the machine, you must first use either the Set Site or the Define Site command.

If you have a Symbolics Ivory-based machine, and you boot a distribution world (the base world that Symbolics sends you), the Command Processor (CP) Show Herald command displays the machine's name as *MACIVORY*, *XL400*, or *UX-family* and the site name as *STANDALONE*.

- If the MacIvory is the only Symbolics machine at your site, you need not use the Set Site or Define Site commands.
- If the MacIvory is one of multiple Symbolics machines at your site, you must use either the Set Site or the Define Site command. To do so:

1. Double-click on the Genera icon to boot the Genera application. See the section "Using the Genera Application on a MacIvory".

2. Edit the `hello.boot` file and remove the Set Boot Options command line from the file.

3. Add the Set Network Address command line to the `hello.boot` file, with a Chaos address for your machine. See the section "Set Network Address FEP Command".

4. Cold boot the MacIvory and use the Set Site or Define Site command.

Here are the criteria for determining whether to use the Set Site or Define Site command on a Symbolics 3600-family or Ivory-based machine:

Set Site	If the namespace files have already been created, use the Set Site comand. This gives your machine access to the already configured site's resources.
Define Site	If the namespace files do not yet exist, use the Define Site command. This creates a new namespace that you can expand later to include other hosts, sites, users, printers, and networks.

Set Site **Command**

Set Site *site-name*

Configures the local distribution world to be an already existing site.

site-name {*name*, get-from-network} The name of your site.

Any further arguments are entered through an AVV menu that adjusts depending on the parameters needed. The Set Site command also fully supports multiple sites within one namespace so the site name does not have to match the namespace name, although one site in any namespace must have a name that is the same as the namespace name.

If the Set Site command is used when the local machine is already registered in a site, the current site is changed to the distribution site before changing over to the new site. This is to eliminate any problems with dangling references to the previous site.

The Set Site command enables your machine to identify all objects included in the site's namespace database. The namespace database for each site is stored in the file system accessible from a machine called the namespace server.

In order for the Set Site command to work, your machine must be a registered host in the site's namespace. If the site's namespace server is not the local host, you must know the namespace server's name and network address.

See the section "Set Site Dialogue".

Define Site **Command**

Define Site *site-name*

Defines a new site.

Type the Define Site command immediately after you boot a new distribution world when you want to define a new site and namespace; it brings up a menu to create a new namespace called *site-name*; when you start this dialogue the local host is, by default, the site's:

- Primary namespace server.
- SYS host.
- Host for storing namespace database files.
- Host for bug reports.

If you want non-local host(s) to perform any of these jobs, provide their primary network addresses and operating system types in the appropriate menu slots.

During the Define Site dialogue, the namespace database files (object files, log files, changes files, and a descriptor file) are created for you on the file system of the machine you specify as the namespace server. Make sure the file system exists on a host accessible to the namespace server machine before you issue the Define Site command. For more information about the namespace database files, see the section "Namespace Database Files".

The namespace server for the new site will have these initial default attributes when the Define Site command is used (*nnnnn* represents a valid octal Chaos address):

```
System Type*: LISPM
Machine Type: 3600
Service: CHAOS-STATUS CHAOS-SIMPLE CHAOS-STATUS
Service: SHOW-USERS CHAOS NAME
Service: TIME CHAOS-SIMPLE TIME-SIMPLE
Service: UPTIME CHAOS-SIMPLE UPTIME-SIMPLE
Service: LOGIN CHAOS TELNET
Service: LOGIN CHAOS SUPDUP
Service: LOGIN CHAOS 3600-LOGIN
Service: SEND CHAOS CONVERSE
Service: SEND CHAOS SEND
Service: NAMESPACE CHAOS NAMESPACE
Service: NAMESPACE-TIMESTAMP CHAOS-SIMPLE NAMESPACE-TIMESTAMP
Service: LISPM-FINGER CHAOS-SIMPLE LISPM-FINGER
Service: FILE CHAOS NFILE
Service: FILE CHAOS QFILE
Service: CONFIGURATION CHAOS CONFIGURATION
Address: CHAOS 12345
```

See the section "Define Site Dialogue".

Introduction to the Namespace Database

A namespace database contains objects. An object must belong to one (and only one) of seven classes in order to be registered in the namespace database:

Host	Represents any computer, usually connected to a network.
User	Represents a person who uses any of the hosts, or a daemon user.
Network	Represents a computer network, to which some hosts are attached.
Printer	Represents a device for producing hardcopy.
Site	Represents a collection of hosts, printers, and networks, grouped together in one location.
Namespace	A database containing information about the mappings from object names to objects, and about the objects themselves.
File System	Reserved for Statice.

Objects in the namespace database have global-names (which identify them), and attributes (which describe them).

Data Types for Attributes

The following data types can be used with attributes in the namespace database:

Global-Name	A name that is shared by all namespaces.
Token	An arbitrary character string.
Set	One or more elements of the same data type.
Pair	Two elements; each element can be of a different data type.
Triple	Three elements; each element can be of a different data type.

Global-name and *token* require you to supply one value. *Set*, *pair* and *triple* require you to supply compound (one or more, two, or three) values. For more information about attributes, see the section "Attributes for Objects in the Namespace Database".

Use the Namespace Editor to "create" (or document) objects, their global-names, and their attributes. For more information about the Namespace Editor, see the section "Using the Namespace Editor".

Names and Namespaces

Every object has a *name*, which is a character string. Two objects of different classes can have the same name. For example, there can be a printer named George and a user named George. An object is identified both by its class and its name.

This means that if you want to look up an object in the database (and you know its name) you have to say "Find the printer named George" or "Find the user named George". You cannot say "Find George".

A namespace is a database that contains mappings from names to objects. Names in one namespace are unrelated to names in another namespace. Specifically, a namespace maps from [class, name] pairs to objects, since every object is identified both by its class and by its name.

Typical sites have one namespace, and the names of all the objects at that site are in that namespace. An object in some namespace other than your own can be referred to by a *qualified name*, which consists of the name of the namespace, a vertical bar, and the name of the object in that namespace.

When long-distance networks link together different sites, the possibility for name conflicts arises. Neither site is forced to change its names just because it wants to connect to the other site.

For example, suppose both Harvard and Yale have computer centers. Harvard has three hosts named Yellow, Orange, and Blue. Yale has three hosts named Apple, Orange, and Banana. Each computer center has its own namespace; Harvard's is named Harvard and Yale's is named Yale.

At Harvard, the Harvard computers are referred to by their unqualified names (Yellow, Orange, and Blue), but the Yale computers are referred to by their qualified names (Yale|Apple, Yale|Orange, and Yale|Banana). At Yale, it would all work the other way around.

Each namespace also has a list of namespaces for its *search rules*. When a name is looked up, each of the *search rules* namespaces listed is consulted in turn, until an object of the desired name is found in one of them. If you list namespaces other than your own in your *search rules*, it is easier to refer to objects in those namespaces, because you do not need to use qualified names for them (unless a name conflict exists).

For example, in the scenario above, the *search rules* for the Harvard namespace could list the Harvard namespace first and the Yale namespace second. Then, users at Harvard could refer to Yale's computers as Apple, Yale|Orange, and Banana. The qualified name Yale|Orange is only necessary because a name conflict exists.

Actually, only some classes of objects have names that are in namespaces; other classes of objects are *globally* named, which means that their names are universal, and conflicts are not permitted. In particular, namespaces and sites are globally named; networks, hosts, printers, and users are not (instead, they're named within namespaces).

Some namespaces do not correspond to any local site. Most large nationwide or worldwide networks have their own host-naming convention. For example, the U.S. Department of Defense Internet has its own set of host names, and this is considered a namespace. If a local site includes some hosts that are on the Internet, it might want to put the Internet namespace into its search list, and install gateways to access Internet machines. For more information, see the section "Namespace Editor CP Commands".

Some objects can also have *nicknames*. In particular, networks and hosts can have nicknames; objects of other classes cannot. A nickname serves as an alternative name for the object. Sometimes you give an object a nickname because its full name is too long to type conveniently. However, each object only has one primary name.

It is possible for an object to be in several namespaces at once. For example, a host which is on both the Internet and a local network at some site might be in both the Internet namespace and the local namespace. In this case, both namespaces maintain their own separate information on the object. The information from each namespace is merged before being presented to the user.

Note: Search lists are not followed recursively. If a user at Harvard looks up a name and Yale's namespace is in Harvard's search list, Yale's search list is not followed.

Using the Namespace Editor

You use the Namespace Editor to register new users and new hardware in the namespace database. To do so, you create and save namespace objects representing the new addition to the site. Once an object has been globally saved, it becomes part of your site's configuration.

The Namespace Editor checks input, and supplies both help and completion. Specifically, the Namespace Editor:

- Checks for errors in network addresses.
- Verifies the nicknames in use by other hosts in the local namespace.
- Checks for unknown services, mediums, and protocols.

You can reach the Namespace Editor in these ways:

- Use the Command Processor (CP) Select Activity command (and select the Namespace Editor Activity). See the section "Workbook: Selecting a New Activity".
- Assign the Namespace Editor to a SELECT key combination with the Select Key Selector or the **tv:add-select-key** function. For more information about how to use the SELECT key, see the section "Customizing the SELECT Key".
- Invoke individual Command Processor (CP) Namespace Editor commands in a Lisp Listener. See the section "Namespace Editor CP Commands".

Creating a New Namespace Object

First select the Namespace editor by using the Edit Namespace Object command. To create a new namespace object, click on [Create Object]. You are prompted for the class and the name of the new object. A template for the information is displayed in the top window. The attributes are mouse sensitive. Clicking on an attribute prompts you in the bottom window for the information to put in the attribute.

Note that the required attributes appear with an asterisk (*) after them. All object classes have a small number of required attributes, and several optional attributes.

You can also create a new object by copying an existing object. Enter Copy Object at the Namespace Editor prompt. Alternatively, use Create Object with the :Copy From keyword.

The window can be scrolled using the SCROLL and m-SCROLL keys or with the mouse. See the section "Scrolling with the Mouse".

When you are satisfied with the information, you can enter it in the database by clicking on [Save Object]. Then click on [Quit] to exit the namespace editor.

For a discussion of saving (locally or globally) new information in the namespace database: See the section "Editing a Namespace Object".

Editing a Namespace Object

Select the Namespace Editor by using the Edit Namespace Object command. If you do not supply the class and object name to Edit Namespace Object, the Namespace Editor window comes up empty and you can click on [Edit Object] or enter the Edit Object command. You are prompted for the name of an object to edit. The current information for the object is retrieved from the namespace database and displayed in the window.

Click Middle on the attribute name for information on the attribute.

The attribute fields are mouse-sensitive. Clicking on an attribute prompts you for information. Mouse clicks have the following meaning:

Left Replace the information in the attribute.

Middle Edit information in the attribute.

Right Menu.

sh-Middle Delete the information in the attribute.

The window can be scrolled using the SCROLL and m-SCROLL keys, the scroll bar, or the mouse.

Once you have finished editing the information, you have three ways to proceed. You can click on [Quit] without saving the changed information. If you are just practicing using the Namespace Editor, that would be appropriate.

The other two choices are to save the information locally or globally. If you save it globally, the new information is stored in the site's namespace database. If you save it locally, the new information is stored only in your machine's local copy of the namespace; changes saved locally affect only your machine (until it is rebooted).

The initial state of the Namespace Editor is the global mode. When you are in global mode the middle of the screen looks like:

Editing HOST TOWHEE in namespace SCRC

If you have clicked on [Locally], you are in local mode. The middle of the screen looks like:

Editing HOST TOWHEE (locally) in namespace SCRC

You can click on [Locally] to toggle the mode between global and local. When you are ready, click on [Save Object] to save the information. Then click on [Quit] to exit the Namespace Editor.

For a complete list of the namespace editor commands, see the section "Namespace Editor Commands".

Namespace Editor Commands

Commands in the Namespace Editor are available as menu items, or as commands you type to the Namespace Editor prompt. This section first gives a short-form summary about each Namespace Editor command, and then describes each command in more detail.

Command Name	*Command Description*
Clear History	Clears objects previously read into the Namespace Editor.
Copy Object	Creates a new namespace object that's similar to an existing one (use the one you've copied as a starting point).
Create Object	Creates a new namespace object.
Delete Object	Deletes a namespace object.
Edit Object	Reads the description of an existing object into the Namespace Editor.
Help	Displays help about the (Namespace Editor-related) topic you specify.
Locally	Toggles whether changes are to be made either to the current world or to the site's namespace database.
Not Modified	Marks the object "not modified". Also bound to `m-~` .
Previous Object	Returns to the previously edited object. Also bound to `c-m-L` (which can take a numeric argument).
Revert Object	Reverts an object's description to the state it was in prior to the current editing session.
Save Object	Saves the (edited) description of an object.
Show History	Shows all the objects that have been read into the Namespace Editor. Also bound to `c-0 c-m-L`.

Namespace Editor Command: Clear History

Clear History

Clears the history of objects from the Namespace Editor. After issuing this command, there is no current object in the Namespace Editor, and no history will exist from which to select a Previous Object. See the section "The Show History Namespace Editor Command".

Namespace Editor Command: Copy Object

Copy Object *name keywords*

Creates a new namespace object named *name* of the same type as the current object. The initial attributes for the new object (except short names and nicknames) are copied from the current object. Once you have copied an object, you can edit it and save the namespace information about the new object.

name The name of the object to create.

keywords :Insert Defaults, :Locally

:Insert Defaults Toggles the Namespace Editor insert-defaults mode on or off for the current namespace object. The initial mode is off.

When the insert-defaults mode is enabled for an object, the Namespace Editor inserts default values to the following indicators:

File Control Lifetime
Home Host
Lispm Name
Mail Address
Pretty-Name
Printer and Bitmap Printer
Printer Interface
Services
Site

See the section "The Insert Defaults Namespace Editor Command" for details on how default values are derived.

:Locally {Yes, No} The initial default is No, which updates the global namespace database (the namespace server) to reflect the changes you have made. Answer Yes to update only the local Lisp world with the changes you have made; the changes will be "forgotten" when you cold boot the local machine (unless you save the changed world). The default for this keyword is displayed along with the object's name, and, if you are editing multiple objects in the Namespace Editor, the current setting becomes the default for subsequent objects.

Namespace Editor Command: Create Object

Create Object *class name keywords*

Creates a new namespace object of type *class* named *name* and makes it the current object in the Namespace Editor. Once the object has been created, you can edit and save it.

class The class of the object to create, such as Host, Printer, User, Site, Namespace, and so on.

name The name of the object to create.

keywords :Copy From, :Insert Defaults, :Locally, :Property-list

:Copy From {*name of an object of the same class*} This will copy the contents of the object named into the newly created object. Duplicate names are removed before the contents of the existing object are inserted into the new object.

:Insert Defaults Toggles the Namespace Editor insert-defaults mode on or off for the current namespace object. The initial mode is off.

When the insert-defaults mode is enabled for an object, the Namespace Editor inserts default values to the following indicators:

File Control Lifetime
Home Host
Lispm Name
Mail Address
Pretty-Name
Printer and Bitmap Printer
Printer Interface
Services
Site

See the section "The Insert Defaults Namespace Editor Command" for details on how default values are derived.

:Locally {Yes, No} The initial default is No, which updates the global namespace database (the namespace server) to reflect the changes you have made. Answer Yes to update only the local Lisp world with the changes you have made; the changes will be "forgotten" when you cold boot the local machine (unless you save the changed world). The default for this keyword is displayed along with the object's name, and, if you are editing multiple objects in the Namespace Editor, the current setting becomes the default for subsequent objects.

:Property-list {*Lisp expression*} This specifies an initial property list for the newly created object. This keyword option will have no effect if the :Copy From keyword option is used.

Namespace Editor Command: Delete Object

Delete Object

Deletes the current object from the namespace database. If the current object is being edited locally, the deletion only happens in Lisp virtual memory, not in the (global) namespace database. This command prompts you for confirmation.

Namespace Editor Command: Edit Object

Edit Object *namespace-object keywords*

Reads in an object from a namespace server, makes it the current object, and allows you to make changes to it.

namespace-object A namespace object is specified by the class of the object followed by the name of the object. For instance, to specify the printer named Asahi, you enter:

```
Edit Object printer Asahi
```

If *namespace-object* is described in more than one namespace (is "multi-homed"), Edit Object prompts for the namespace in which you want to edit this object. (Typically, a namespace object is described only in one namespace.)

Keywords :Insert Defaults, :Locally

:Insert Defaults Toggles the Namespace Editor insert-defaults mode on or off for the current namespace object. The initial mode is off.

When the insert-defaults mode is enabled for an object, the Namespace Editor inserts default values to the following indicators:

File Control Lifetime
Home Host
Lispm Name
Mail Address
Pretty-Name
Printer and Bitmap Printer
Printer Interface
Services
Site

See the section "The Insert Defaults Namespace Editor Command" for details on how default values are derived.

:Locally {Yes, No} The initial default is No, which updates the global namespace database (the namespace server) to reflect the changes you have made. Answer Yes to update only the local Lisp world with the changes you have made; the changes will be "forgotten" when you cold boot the local machine (unless you save the changed world).

The default for this keyword is displayed along with the object's name, and, if you are editing multiple objects in the Namespace Editor, the current setting becomes the default for subsequent objects.

Namespace Editor Command: Help

Help *topic*

Displays help about a namespace-related topic.

topic The topics include Namespace Editor commands and menu items, overviews, and object attributes.

Namespace Editor Command: Insert Defaults

Insert Defaults

Toggles the insert-defaults mode on and off for the current Namespace Editor object (the initial mode is off). Note that the insert-defaults mode is not "sticky". That is, if it is enabled for the current namespace object, it is not enabled for new objects read into the Namespace Editor (unless explicity specified).

When the insert-defaults mode is enabled, the Namespace Editor inserts default indicator values as follows:

File Control Lifetime
: The value of **fs:*default-file-control-connection-lifetime*** is used (30 minutes).

Home Host
: Defaults to the host in the mail address when it is available.

Lispm Name
: Defaults to the user object name.

Mail Address
: Defaults to User-name Home-Host when home-host is available.

Pretty-Name
: The pretty-name defaults to the string-capitalized name of the object.

Printer and Bitmap Printer
: The site's default printers are used.

Printer Interface
: Defaults to serial.

Services
: When a system-type and network are available, services from **neti:*supported-services*** are added to the object.

Site
: For hosts and printers, the site defaults to the site with the same name as the namespace for the object.

Note that, if you are adding default Services to a namespace server at a site where pre-Genera 8.0 machines are still running, the TCP namespace services that are added can cause problems for booting pre-Genera 8.0 machines.

You can also use the :Insert Defaults keyword for the following commands to enable/disable the insert-defaults mode for a namespace object:

"The Edit Object Namespace Editor Command"
"The Create Object Namespace Editor Command"
"The Copy Object Namespace Editor Command"
"Edit Namespace Object Command"
"Create Namespace Object Command"

Namespace Editor Command: Locally

Locally

Toggles whether the current object is being edited locally or globally.

If you save an object locally, the namespace database on the namespace server is not changed. Only the local machine sees the changes; the changes are lost when you cold boot (unless you have saved the changed world).

If you save an object globally (the default in most cases), the namespace database on the namespace server is changed. This makes the new namespace information available to the entire site.

Namespace Editor Command: Not Modified

Not Modified

Changes the current object's status to "not modified". This command is bound to m-~.

See the section "The Save Object Namespace Editor Command".

Namespace Editor Command: Previous Object

Previous Object *keyword*

Selects the previous object read into the Namespace Editor, and makes it the current object. This command is bound to c-m-L.

keyword	:Count
:Count	{*integer*} Means go back this many objects on the history list. The default is 2. Alternatively, a numeric argument can be given to c-m-L.

See the section "The Show History Namespace Editor Command".

Namespace Editor Command: Revert Object

Revert Object

Discards any recently made changes to the current object. If the object is being edited locally, it reverts to the version in the local machine's virtual memory. If the object is being edited globally, it reads a fresh copy of the object in from the namespace server.

Namespace Editor Command: Save Object

Save Object *keyword*

Saves the current object. If the object is being edited locally, this command saves the changes only to the local machine's Lisp world. If the object is being edited globally, this command saves the changes to the global namespace database (on the namespace server).

keyword :Force Save

:Force-save {Yes, No} The default is No, which means do nothing if no changes were made. A Yes answer saves the object even if no changes were made (or if the Not Modified namespace editor command was used).

Namespace Editor Command: Show History

Show History

Displays a mouse-sensitive list of all the objects read into the Namespace Editor. This is bound to `c-0 c-m-L`.

Here is an example of the display:

```
1.  PRINTER PRENSA in namespace SCRC.
2.  NETWORK INTERNET (locally) in namespace SCRC.
3.  HOST WATERFOWL (locally) in namespace SCRC.
4.  USER CRAWLEY in namespace SCRC.
```

See the section "The Previous Object Namespace Editor Command".

Namespace Editor CP Commands

This section describes Command Processor (CP) commands related to the Namespace Editor.

Add Services to Hosts Command

Add Services to Hosts *service-triples specific-hosts-or-all-hosts keywords*

Adds the specified service attributes in the namespace for one or more hosts.

service-triples A service triple consists of a service, a medium, and a protocol. For more information, see the section "Concepts of Service, Medium, and Protocol".

specific-hosts-or-all A list of hosts (or "all") to which you want to add services. If you specify all, use the :Namespace, :Site, and :Type keywords.

Keywords:

:Locally {Yes, No} The initial default is No, which updates the global namespace database (the namespace server) to reflect the changes you have made. Answer Yes to update only the local Lisp world with the changes you have made; the changes will be "forgotten" when you cold boot the local machine (unless you save the changed world).

The default for this keyword is displayed along with the object's name, and, if you are editing multiple objects in the Namespace Editor, the current setting becomes the default for subsequent objects.

:Verbose Prints messages for each host modified.

:Namespace Adds the services to all hosts in the namespace (use with the "all" argument).

:Site Adds the service to all hosts in the site (use with the "all" argument).

:Type Adds the service to all hosts of this system type (use with the "all" argument).

Here is an example adding netboot service to three hosts at once:

```
Command: Add Services To Hosts (A Service Triple) (service) netboot
(medium) slap (protocol) netboot
(A sequence of hosts or All) HARPAGORNIS, WINTER, TOWHEE
(keywords) :Locally

Adding service NETBOOT SLAP NETBOOT to hosts (locally).
Done.
```

Create Namespace Object **Command**

Create Namespace Object *class name keywords*

Adds a new object to the namespace database. For more information about adding objects to the namespace database, see the section "Creating a New Namespace Object".

class	{File-System, Host, Namespace, Network, Printer, Site, User} The type of object you want to create.
name	The name of the new object.
Keywords	:Copy From, :Insert Defaults, :Locally, :Property-List
:Copy From	{name of an object of the same class} This will copy the contents of the object named into the newly created object. Duplicate names are removed before the contents are inserted into the new object to avoid conflicts.
:Insert Defaults	Toggles the Namespace Editor insert-defaults mode on or off for the current namespace object. The initial mode is off. When the insert-defaults mode is enabled for an object, the Namespace Editor inserts default values to the following indicators: File Control Lifetime Home Host Lispm Name Mail Address Pretty-Name Printer and Bitmap Printer Printer Interface Services Site See the section "The Insert Defaults Namespace Editor Command" for details on how default values are derived.
:Locally	{Yes, No} The initial default is No, which updates the global namespace database (the namespace server) to reflect the changes you have made. Answer Yes to update only the local Lisp world with the changes you have made; the changes will be "forgotten" when you cold boot the local machine (unless you save the changed world). The default for this keyword is displayed along with the object's name, and, if you are editing multiple objects in the Namespace Editor, the current setting becomes the default for subsequent objects.
:Property List	{a Lisp expression} This specifies an initial property list for the newly created object in the internal form that object properties are specified in. This keyword option will have no effect if the :Copy From keyword option is used.

Delete Namespace Object **Command**

Delete Namespace Object *class name keywords*

Removes information about the object *name* from the namespace.

class {File-System, Host, Namespace, Network, Printer, Site, User} The type of object you want to delete.

name The name of the object you want to delete.

Keywords: :Locally

:Locally {Yes, No} The initial default is No, which updates the global namespace database (the namespace server) to reflect the changes you have made. Answer Yes to update only the local Lisp world with the changes you have made; the changes will be "forgotten" when you cold boot the local machine (unless you save the changed world). The default for this keyword is displayed along with the object's name, and, if you are editing multiple objects in the Namespace Editor, the current setting becomes the default for subsequent objects.

If the object you are deleting is still referenced by other objects, for example, if you are deleting a host and a user still lists that host as a mail address, you cannot delete it. This protects your namespace from becoming corrupted. You get an error telling you which objects still reference the object you are trying to delete so you can edit those objects to remove the references.

```
Error: Error from Namespace on RIVERSIDE: You cannot delete HOST QUABBIN.
       It is still referenced by the following objects:
         SITE SCRC             USER LISP-MACHINE  USER WOBBLY
         USER SCH|FILE-SERVER  USER DCP           USER HASEGAWA
         USER NFEP             USER FILE-SERVER
         USER NISHIMOTO        USER ZIPPY
```

Edit Namespace Object **Command**

Edit Namespace Object *namespace-object keywords*

Modifies an object in the namespace database.

To create a new object, see the section "Create Namespace Object Command".

namespace-object A namespace object is specified by the class of the object followed by the name of the object. For instance, to specify the printer named Asahi, you enter:

```
Edit Namespace Object printer Asahi
```

If *namespace-object* is described in more than one namespace (is "multi-homed"), Edit Namespace Object prompts for the namespace in which you want to edit this object. (Typically, a namespace object is described only in one namespace.)

Keywords :Insert Defaults, :Locally

:Insert Defaults Toggles the Namespace Editor insert-defaults mode on or off for the current namespace object. The initial mode is off.

When the insert-defaults mode is enabled for an object, the Namespace Editor inserts default values to the following indicators:

File Control Lifetime
Home Host
Lispm Name
Mail Address
Pretty-Name
Printer and Bitmap Printer
Printer Interface
Services
Site

See the section "The Insert Defaults Namespace Editor Command" for details on how default values are derived.

:Locally {Yes, No} The initial default is No, which updates the global namespace database (the namespace server) to reflect the changes you have made. Answer Yes to update only the local Lisp world with the changes you have made; the changes will be "forgotten" when you cold boot the local machine (unless you save the changed world).

The default for this keyword is displayed along with the object's name, and, if you are editing multiple objects in the Namespace Editor, the current setting becomes the default for subsequent objects.

Remove Service**s** From Hosts **Command**

Remove Services from Hosts *service-triples specific-hosts-or-all keywords*

Removes the specified service attributes in the namespace for one or more hosts.

service-triples A service triple consists of a service, a medium, and a protocol. For more information, see the section "Concepts of Service, Medium, and Protocol".

specific-hosts-or-all A list of hosts (or "all") from which you want to remove service. If you specify all, use the :Namespace, :Site, and :Type keywords.

keywords: :Locally, :Verbose, :Namespace, :Site, :Type

:Locally — {Yes, No} The initial default is No, which updates the global namespace database (the namespace server) to reflect the changes you have made. Answer Yes to update only the local Lisp world with the changes you have made; the changes will be "forgotten" when you cold boot the local machine (unless you save the changed world).

The default for this keyword is displayed along with the object's name, and, if you are editing multiple objects in the Namespace Editor, the current setting becomes the default for subsequent objects.

:Verbose — Print messages for each host modified.

:Namespace — Add the services to all hosts in the namespace (use with the "all" argument).

:Site — Add the services to all hosts in the site (use with the "all" argument).

:Type — Add the services to all hosts of this system type (use with the "all" argument).

Here is an example of removing netboot service from a host:

```
Remove Services From Hosts (Service Triples (service) netboot
(medium) slap (protocol) netboot
(A sequence of hosts or All) HARPAGORNIS

Removing service NETBOOT SLAP NETBOOT from hosts.
Done.
```

Show Namespace Object **Command**

Show Namespace Object *namespace-object keywords*

Shows the information in the namespace database.

namespace-object — A namespace object is specified by the class of the object followed by the name of the object. For instance, to specify the printer named Asahi, you enter:

```
Show Namespace Object printer Asahi
```

If *namespace-object* is described in more than one namespace (is "multi-homed"), Edit Namespace Object prompts for the namespace in which you want to edit this object. (Typically, a namespace object is described only in one namespace.)

keywords: — :Format, :Locally, :More Processing, :Output Destination

:Format — {Normal, Detailed} Whether to show fields that are empty. The default is normal, to omit empty fields. Detailed shows all fields.

:Locally — Specifies whether the Show Namespace Object command queries the namespace server or uses the values in the current Lisp world.

:More Processing — {Default, Yes, No} Controls whether **More** processing at end of page is enabled during the output to interactive streams. The default is Default.

If No, output from this command is not subject to **More** processing.

If Default, output from this command is subject to the prevailing setting of **More** processing for the window. (See the section "FUNCTION M".)

If Yes, output from this command is subject to **More** processing unless it was disabled globally (see the section "FUNCTION M").

:Output Destination — {Buffer, File, Kill Ring, Printer, Stream, Window} Where to redirect the typeout done by this command. The default is the stream ***standard-output***.

Here is what the namespace object for a user might look like:

Showing USER KJONES in namespace ACME:

```
Lispm Name: Kjones
Personal Name: Jones, Kingsley
Nickname: King
Work Address: Building 3-701
Work Phone: 5891
Home Host: ACME
Mail Address: Kjones ACME
Birthday: 19 June
Project: Database
Supervisor: Finklestein
```

Namespace Database Files

This section describes the namespace database files. While this information is useful for anyone who wants to learn about namespace database functionality, it's most important for site administrators and maintainers (people who manage the namespace database).

Four types of text files exist to hold information relevant to the namespace database:

1. Descriptor Files.

2. Object Files.

3. Log Files.

4. Changes Files.

Namespace database files are stored on namespace servers. The Namespace Editor provides the interface between you and the namespace database files; when you use the Namespace Editor, it updates each of the namespace database files appropriately.

Namespace database files should *never* be changed by hand. By only using the Namespace Editor to update the namespace database, you ensure the integrity of the namespace database files at your site.

Namespace database files contain records. Records provide printed representations of each object (and its attributes) in the namespace database.

A record is a set of lines. Lines, in turn, are sets of tokens, separated by spaces. A token is a sequence of characters excluding the space, newline, semicolon, and double quote characters, or any sequence of characters located between double quotes. (You can quote within double-quotes by using the backslash character.) A blank line follows every record.

There is a one-to-one relationship between objects in the namespace database and records in the namespace database files; each record describes an object. The first line of a record always consists of the object's class, and its primary name. Subsequent lines contain an attribute indicator and value. (Similar to a property list, the first token in each line is called an *indicator*; other tokens in the line are called *values*.)

Namespace Database Descriptor Files

Each namespace has one descriptor file. Its pathname appears in the Descriptor-File namespace object attribute. A descriptor file holds information that describes where all of the namespace database (object-specific) information is stored.

Each line of the file contains a comment or an indicator, followed by a pathname. Valid indicators are class (object type), version, changes, or asterisk (*).

Table ! describes each indicator and its value.
Here are the sample contents of a descriptor file:

```
;-*-Text-*-
VERSION BLUE:>SYS>SITE>HARVARD-NAMESPACE-LOG.TEXT
CHANGES BLUE:>SYS>SITE>HARVARD-NAMESPACE-CHANGES.TEXT
HOST BLUE:>SYS>SITE>HARVARD-HOSTS.TEXT
USER BLUE:>SYS>SITE>HARVARD-USERS.TEXT
* BLUE:>SYS>SITE>HARVARD-OTHERS.TEXT
```

Indicator	*Value*
Class	The pathname is for a file containing objects of this class.
Version	The pathname is for the log file.
Changes	The pathname is for the changes file.
*	The pathname is for a file that contains objects in other classes (not explicitly named by the indicator *class*).

Table 3. Descriptor File Indicators and Values

Namespace Database Object Files

Object files contain the most current descriptions of all objects that exist within the namespace. An object file begins with a list that specifies the namespace to which it belongs. This is followed by a series of records. Each record is separated by a blank line. For information about records, see the section "Namespace Database Files".

Here are the sample contents from an object file:

```
;-*- Mode: Text; Network-Namespace: Harvard -*-
USER GEORGE
LISPM-NAME George
PERSONAL-NAME "Washington, George"
HOME-HOST BLUE
MAIL-ADDRESS George BLUE
LOGIN-NAME George BLUE
LOGIN-NAME Washington.States MIT|MULTICS
LOGIN-NAME GW MIT|MC
NICKNAME Georgie
WORK-ADDRESS "The White House, Washington D.C., 10001"
WORK-PHONE 202-555-1212
HOME-ADDRESS "Mount Vernon VA"
HOME-PHONE 202-999-1234
PROJECT "being President of the United States"
SUPERVISOR "the People"
REMARKS "I cannot tell a lie."
```

Namespace Database Log Files

Namespace database log files contain all of the changes made to a database. A namespace database log file's file-system version number serves as a timestamp for the change that resulted in that version being written out. This timestamp helps the namespace database to identify obsolete data.

Here are the sample contents from a log file:

```
1/24/90 16:39:22 USER GEORGE by George.  Old timestamp was 607.
1/24/90 22:09:10 HOST BLUE by JAdams.  Old timestamp was 608.
1/26/90 07:23:45 HOST GREEN deleted by JAdams.
```

Namespace Database Changes Files

Namespace database changes files provide a chronological record of all the changes to the namespace database. This enables hosts to process only those changes to the namespace database that have occurred since the last time the world was saved on the namespace server.

Each entry in the changes file consists of:

- A timestamp line.
- Lines for deleted objects (optional).
- A blank line.
- Changed or added object records (optional).

The timestamp line consists of the word "timestamp", followed by the version number of the log file before the change was made. Deleted objects are identified by their class and primary name. Changed objects appear just as they do in the object file.

If an object is changed twice, only the newest record for it will appear. Older entries in the file are likely to consist only of a timestamp line and a blank line. Run **neti:prune-namespace-changes-file** to excise them.

Here are the sample contents of a changes file:

```
TIMESTAMP 608
HOST BLUE
SYSTEM-TYPE LISPM
SERVICE CHAOS-STATUS CHAOS-SIMPLE CHAOS-STATUS
SERVICE SHOW-USERS CHAOS NAME
SERVICE TIME CHAOS-SIMPLE TIME-SIMPLE
SERVICE UPTIME CHAOS-SIMPLE UPTIME-SIMPLE
SERVICE LOGIN CHAOS TELNET
SERVICE SEND CHAOS SEND
SERVICE MAIL-TO-USER CHAOS CHAOS-MAIL
SERVICE NAMESPACE CHAOS NAMESPACE
SERVICE NAMESPACE-TIMESTAMP CHAOS-SIMPLE NAMESPACE-TIMESTAMP
SERVICE LISPM-FINGER CHAOS-SIMPLE LISPM-FINGER
SERVICE FILE CHAOS QFILE
LOCATION Kiosk 1
FINGER-LOCATION "Harvard Square Kiosk"
PRETTY-NAME Yellow
ADDRESS CHAOS 24412
MACHINE-TYPE LISPM
NICKNAME YEL
SHORT-NAME Y
SITE HARVARD

TIMESTAMP 609
HOST GREEN

TIMESTAMP 610
```

Attributes for Objects in the Namespace Database

Objects in the namespace database can have attributes associated with them. Attributes describe the characteristics of an object. Like Lisp property lists, attributes have:

1. An *indicator* (the name of the attribute).

2. A *value*.

Although objects have one or more required attributes, most attributes are optional. For example, every host object has a System-Type attribute (this describes the operating system it runs), and every printer object has a Type attribute (this describes the kind of printer it is). Host objects can optionally have a Pretty-Name attribute, printer objects can optionally have a Default-Font attribute, and user objects can optionally have a Home-Phone attribute associated with them.

Some attributes can be associated more than once with a given object. For example, hosts can have multiple Address attributes if they are attached to multiple networks.

When editing a namespace object, it's easy to determine whether its attributes are required or optional; required attributes are marked with an asterisk. Remember that each class of objects has a prescribed set of both required and optional attributes (you cannot create additional attributes for the objects within your namespace database). For more information about objects in the namespace database, see the section "Introduction to the Namespace Database".

Attributes for Objects of Class "Namespace"

These attributes belong to objects of class "namespace".

In the Namespace Editor, required namespace object attributes have an asterisk by them.

Namespace Object Attribute: Descriptor-File

Descriptor-File
: A pathname that specifies the descriptor file for a namespace (descriptor files say where all of the objects in a namespace database are stored). For example:

```
Descriptor File*: BLUE:>SYS>SITE>HRV-NAMESPACE.TEXT
```

For more information about namespace database descriptor files, see the section "Namespace Database Descriptor Files".

Namespace Object Attribute: Internet-Domain-Name

Internet-Domain-Name
: Specifies an Internet Domain Name for a namespace. For example,

```
Internet Domain Name: SCRC.Symbolics.COM
```

To avoid naming conflicts, Internet Domain Names for namespaces must be unique (that is, registered with the Network Information Center).

For more information, see the section "Dialnet and Internet Domain Names".

Namespace Object Attribute: Primary-Name-Server

Primary-Name-Server
: The host that is the primary namespace servers for a namespace. Primary servers are the authority regarding a namespace; namespace data are stored in files belonging to the primary namespace server. For example,

```
Primary Name Server: BLUE
```

More information is available about the namespace database files. See the section "Namespace Database Files".

Namespace Object Attribute: Search-Rules

Search-Rules

Namespaces listed in the order that they will be used when searching for an object. Note that the namespace itself must appear in the search rules. For example:

```
Search Rules*: HARVARD YALE BROWN
```

More information is available about search rules. See the section "Names and Namespaces".

Namespace Object Attribute: Secondary-Name-Server

Secondary-Name-Server

Hosts that are secondary namespace servers for a namespace. Secondary namespace servers are not authoritative regarding a namespace. Rather, they provide a backup in case a primary namespace server is unavailable.

Secondary namespace servers attempt to keep a copy of the namespace information current by querying the primary server more often than a nonserver machine would.

Secondary namespace servers check every namespace that the host knows about. So that your secondary namespace server will look at only one other namespace, see the section "The Default Secondary Name Server Host Attribute".

Namespace Object Attribute: User-Property

User-Property

All objects contained within the namespace (hosts, sites, namespaces, printers, and users) are eligible to have a User-Property attribute. It consists of a pair whose first element is an indicator (like that of a property list) and whose second element is a token. The User-Property attribute holds any information that users choose to associate with an object. For example:

```
User-Property: ID-number 123-45-6789
```

Attributes for Objects of Class "Site"

These attributes belong to objects of class "site" within a namespace. Site objects are collections of hosts, printers, and networks located together.

In the namespace editor, required site object attributes have an asterisk by them.

Site Object Attribute: All-Mail-Addresses-Forward

All-Mail-Addresses-Forward

If set to Yes, this option means that Foo@BAR and Foo@QUUX are the same address if BAR and QUUX are in the same site. In particular, it permits any Mailer at the site to lookup the address in its mailbox table and process it appropriately. If set to No, Mailers are forced to deliver the message to whatever host at the site is specified.

Note: Setting All-Mail-Address-Forward to Yes does not interact well with non-Symbolics mailers at the same site.

Site Object Attribute: Default-Bitmap-Printer

Default-Bitmap-Printer

Specifies the default printer for printing screen images at this site; a printer object. This is used by hosts that do not have their own Bitmap-Printer Attribute.

Site Object Attribute: Default-Printer

Default-Printer — Specifies the default printer for printing text files at this site; a printer object. Hosts that do not have their own Printer Attribute use the printer specified by this site attribute.

Site Object Attribute: Dont-Reply-To-Mailing-Lists

Dont-Reply-To-Mailing-Lists

Specifies those mailing lists to which Zmail won't reply to by default; tokens. This attribute is useful only for users who have not set the "people not to reply to" option in their Zmail init files.

Site Object Attribute: Host-for-Bug-Reports

Host-for-Bug-Reports

Specifies the host to which bug reports should be sent (required). The Command Processor (CP) Report Bug command, and other Debugger, Editor, and Zmail bug reporting commands use this attribute.

Site Object Attribute: Host-Protocol-Desirability

Host-Protocol-Desirability

A triple for the Generic Network System's desirability estimates (used when the system tries to construct a path to a service). *Host* represents a host, *protocol* names a protocol that the host supports, and *desirability* is a token expressing a floating-point factor (for the cost calculations). For example:

```
Host Protocol Desirability: YUKON CHAOS-MAIL 0.75
```

Network services and protocols are discussed elsewhere.

- See the section "Symbolics Generic Network System".
- See the section "Desirability of Network Protocols".

If you change the value of **host-protocol-desirability**, you must either cold boot, warm boot, or use the Command Processor (CP) command Reset Network to make the change take effect.

Site Object Attribute: Local-Namespace

Local-Namespace

Specifies the namespace that is local to the site. (Normally, there is exactly one site for each namespace.)

Site Object Attribute: Other-Sites-Ignored-in-Zmail-Summary

Other-Sites-Ignored-in-Zmail-Summary

Specifies a set of site objects whose names will not be displayed in the Zmail summary windows of users at the site.

Site Object Attribute: Other-Sites-in-Mail-Area

Other-Sites-in-Mail-Area

Sites that share the same list of mailboxes as this site. For more information, see the section "Symbolics Store-and-Forward Mailer".

Site Object Attribute: Pretty-Name

Pretty-Name

Most objects contained within the namespace (sites, other namespaces, networks, hosts, and printers) are eligible to have a Pretty-Name attribute. The Pretty-Name attribute specifies a name; a token.

The pretty name is the object name that appears on screen displays and in prompts. That is, an object's pretty name appears where people need to see it, while an object's actual name is used by software.

Site Object Attribute: Root-Domain-Server-Address

Root-Domain-Server-Address

A pair that specifies the network type and an address for the root domain server (used by the Domain Name System). Here are some examples:

```
Root Domain Server Address: INTERNET 10.0.0.51
Root Domain Server Address: INTERNET 10.1.0.17
Root Domain Server Address: INTERNET 128.213.5.17
```

More information is available about the Domain Name System.

- See the section "Introduction to Internet Domain Names".
- See the section "Installing the Internet Domain Names System".

Site Object Attribute: Secure-Subnets

Secure-Subnets

One or more lists that specify networks and the "trusted" subnets. Hosts on the specified subnets are "trusted", which means that they are permitted to invoke servers at the site. This attribute controls the subnet security feature of any servers that use the **:trusted-p** or **:reject-unless-trusted** keywords to **net:define-server**. The following server-side protocols respect the **secure-subnets** attribute: NFILE, QFILE, TCP-FTP, and TFTP.

The first element in each list is a network type, either CHAOS or INTERNET. The second element is one or more subnet numbers. For CHAOS networks, the subnet numbers are represented as octal character strings. Internet subnet/network numbers are represented as decimal character strings. For example:

```
Secure Subnets: CHAOS 1 3 4 5 16 22 24 26 30
Secure Subnets: INTERNET 128.81.0.0
```

If this attribute has no value for INTERNET, then no Internet subnets/networks are trusted. If this attribute has the value "ALL" for INTERNET, then all Internet subnets/networks are trusted.

If this attribute has no value for CHAOS or the value "ALL", then all Chaos subnets are trusted.

Site Object Attribute: Site Directory

Site-Directory — A token that specifies the directory that holds the Symbolics computer system's site-specific files. This provides the translation for the logical pathname sys:site; .

All other site-specific pathname translations are managed by logical pathname translations files.

Site Object Attribute: Site-System

Site-System — A token that specifies the name of a system (as **defsystem** does) that contains software you want shared by all of the Symbolics computers at your site. A notification appears when machines are booted without having loaded the site system.

The Standalone Site Attribute

Standalone — Specifies whether the host at this site is a standalone machine; a token. If Yes, only one host exists at this site and no response to the *who-am-I* network broadcast request at boot time is expected.

If No, or should the attribute not be present, multiple Symbolics computer hosts exist at this site; when one host is booted, another answers its *who-am-I* query.

Site Object Attribute: Terminal-F-Argument

Terminal-F-Argument

Defines how arguments to the FUNCTION F key should work. The first element is a number (a decimal string) or the word *none*. The second element is a global-name (this can be: login,

host, local-lisp-machines, or all-lisp machines). The third element is a list of hosts (if host was the second element).

Global names are described in greater detail here:

Login	The login file computer.
Local-Lisp-Machines	All Symbolics computers at this site.
All-Lisp-Machines	All Symbolics computers on the local network.
Host	A list of hosts appears as the third element.

For example:

```
Terminal F Argument: NONE LOCAL-LISP-MACHINES
Terminal F Argument: 0 ALL-LISP-MACHINES
Terminal F Argument: 1 HOST VIXEN CUPID COMET
Terminal F Argument: 2 LOGIN
```

More information is available about Function-F. See the section "FUNCTION F".

Site Object Attribute: Timezone

Timezone — The timezone at this site; a global-name (required). For example:

```
Timezone*: EST
```

Site Object Attribute: User-Property

User-Property — All objects contained within the namespace (hosts, sites, namespaces, printers, and users) are eligible to have a User-Property attribute. It consists of a pair whose first element is an indicator (like that of a property list) and whose second element is a token. The User-Property attribute holds any information that users choose to associate with an object. For example:

```
User-Property: ID-number 123-45-6789
```

Site Object Attribute: Validate-LMFS-Dump

Validate-LMFS-Dump

If Yes, the LMFS backup dumper validates backup tapes. If No (or if the attribute is not provided), validation is not performed.

Attributes for Objects of Class "Host"

These attributes belong to objects of class "host" within a namespace. A host object is any computer connected to a network.

In the namespace editor, required host object attributes have an asterisk by them.

Host Object Attribute: Address

Address

One or more pairs that specify the network addresses for a host. Networks must be valid network objects (already registered in the namespace). Network addresses must be in the correct format for each network type. For example:

```
Address:  CHAOS 401
```

For information about individual network types and the addressing conventions they use, see the section "Network Addressing".

Host Object Attribute: Bitmap-Printer

Bitmap-Printer

Specifies the default bitmap printer for this host. If this attribute is not provided, the site's Default-Bitmap-Printer attribute is used.

Host Object Attribute: Console-Location

Console-Location

A triple that describes the physical location of a host's console.

Prompts for the building, floor number, and a description of the location. (Use quotes to enter multiword values for building and floor number). For example,

```
Console-location: "Empire State" 107 Near King Kong
```

Host Object Attribute: Default-Secondary-Name-Server

Default-Secondary-Namespace-Server

If Yes, this host acts as a secondary namespace server. Typically, each primary namespace server also acts as a secondary namespace server for other namespaces (if they exist for the site).

There are two ways to create a secondary namespace server:

- List secondary namespace servers in the Secondary-Namespace-Server attribute of an object of class "namespace".
- Set the Default-Secondary-Namespace-Server attribute to Yes in a host object.

See the section "The Secondary Name Server Namespace Attribute".

Host Object Attribute: File-Control-Lifetime

File-Control-Lifetime

A time interval that specifies the lifetime of a file control connection. When a Symbolics computer (client) connects to a file server, the client will automatically close the connection after it has been idle for the specified amount of time. For example:

```
File Control Lifetime: 30 minutes
```

More information is available about time intervals. See the section "Reading and Printing Time Intervals".

Host Object Attribute: Finger-Location

Finger-Location

A token that describes the physical location of a host. For example

```
Finger-location:  Across the alley from the Alamo
```

Note: This attribute is obsolete; use the Console-Location host object attribute instead.

Host Object Attribute: Internet-Domain-Name

Internet-Domain-Name

Use this attribute to explicitly specify the Internet Domain Name for a host.

For example, if the default Internet Domain Name for the host should be:

```
Internet Domain Name: Redwing.SCRC.Symbolics.COM
```

you can provide a different string to the Internet-Domain-Name attribute, like this:

```
Internet Domain Name: Redwing.MIT.EDU
```

Note: Symbolics does not recommend that you use this attribute, as it allows you to violate the implementation restrictions for Internet Domain Names, which can result in unpredictable behavior.

You can use the **neti:*allow-dotted-host-names-in-namespaces*** variable to accommodate host names with dots in them. For more information about this, see the variable **neti:*allow-dotted-host-names-in-namespaces***.

See the section "Dialnet and Internet Domain Names".

Host Object Attribute: Location

Location — A pair of tokens that describes the physical location of a host. The first element in the pair specifies the building (or room) in which the machine resides. The second element specifies the floor. For example:

```
Location: Lab 2
```

Note: This attribute is obsolete; use the Console-Location host object attribute instead.

Host Object Attribute: Machine-Type

Machine-Type — Specifies the hardware type of the machine (as opposed to the operating system, see the section "The System Type Host Attribute"); a global-name. It is not required, but strongly recommended. For example:

```
Machine Type: 3653
```

The Machine-Type is used largely for informational display and determining defaults. For example, a tape operation on an embedding host of Machine-Type SUN-3 offers a default for device of `st0:` instead of `Cart:`. Any value is accepted in this field. However, you should try to use the official name as in the list maintained by the Network Information Center Assigned Numbers RFC. The NIC lists Symbolics machine types as `Symbolics-` followed by the model number. The namespace Machine Type Attribute strips the `Symbolics-` off and uses just the model number. The most common machine types Symbolics

deals with are included here:

Machine Type	Host Attribute Value
Symbolics 3600	3600
Symbolics 3610	3610
Symbolics 3620	3620
Symbolics 3630	3630
Symbolics 3640	3640
Symbolics 3645	3645
Symbolics 3650	3650
Symbolics 3653	3653
Symbolics 3670	3670
Symbolics 3675	3675
Symbolics MacIvory	MACIVORY
Symbolics XL400	XL400
Symbolics XL1200	XL1200
Symbolics UX400	UX400S
Symbolics UX1200S	UX1200S
Sun-3	Sun-3
SPARCstation	Sun-4
Digital VAX	VAX
Digital PDP10	PDP10
Digital PDP11	PDP11
IBM PC	IBMPC
Honeywell DPS-8M	HONEYWELL-DPS-8M
Alto	ALTO
CADR	CADR

(Note: For Sun-*n*, any value is accepted, for example Sun-3/260 is accepted as Sun-3. However, SPARCstation and any other names should be entered as Sun-*modelnumber*.)

Host Object Attribute: Nickname

Nickname — Specifies alternative names for the host. A host can have one or more nicknames. For example:

```
Nickname: Redwing
Nickname: Red
```

Host Object Attribute: Peripheral

Peripheral — Specifies a peripheral device. Click on the peripheral type and you are prompted for the values associated with it.

For example, some XL and MacIvory machines use the Emulex M02 controller, a QIC-11 SCSI tape drive. You need to add a PERIPHERAL entry to the namespace object of any host that uses the Emulex MT02 controller. If this entry is not present, only four of the nine tracks will be used. For example, for a controller at SCSI address 1, the entry should look like this:

```
Peripheral: TAPE UNIT SCSI1 MODEL EMULEX-MT02
```

Host Object Attribute: Pretty-Name

Pretty-Name

Most objects contained within the namespace (sites, other namespaces, networks, hosts, and printers) are eligible to have a Pretty-Name attribute. The Pretty-Name attribute specifies a name; a token.

The pretty name is the object name that appears on screen displays and in prompts. That is, an object's pretty name appears where people need to see it, while an object's actual name is used by software.

Unlike the Name, the Nickname, and the Short-Name attributes, the Pretty-Name attribute cannot be used to find a host.

Host Object Attribute: Print-Spooler-Options

Print-Spooler-Options

One or more pairs of global-names and tokens that specifies the directory where hardcopy requests are stored for each print spooler running on this host. A typical global-name for the Print-Spooler-Options attribute is Home-Directory.

The default for Symbolics computers is local:>print-spooler>. Print spooler options can also be specified for individual printers. For example,

```
Print Spooler Options: Home-directory local:>print-spooler>
```

Host Object Attribute: Printer

Printer

Specifies the preferred printer object for this host. This printer is used by default when files are hardcopied from this host. If this attribute is not provided, the site's Default-Printer attribute is used.

Host Object Attribute: Server-Machine

Server-Machine — Specifies whether the object described is a server machine; a token. If the value is Yes, the host is a server machine. If it is No (the default) this host is not a server machine.

This attribute applies only to Symbolics computers. Server machines do not automatically enable their services when you boot them; you must enable services in the server's lispm-init file, using **sys:enable-services** or the Command Processor (CP) Enable Services command. See the section "Enable Services Command".

Host Object Attribute: Service

Service — A triple specifying that a host is capable of providing *service* when connected via *medium* and *protocol*. For example,

```
Service:  FILE CHAOS NFILE
```

For information on services, mediums, and protocols, see the section "Service Attributes in the Namespace Database".

Host Object Attribute: Short-Name

Short-Name — Specifies a set of short-names to be used when a program wants to display a host's name without taking up much space.

Host Object Attribute: Site

Site — Specifies the site (a site object) at which this host is located (required). For example,

```
Site*: SCRC
```

Host Object Attribute: Spooled-Printer

Spooled-Printer — Specifies printers for which this host provides a spooling service. When you enter a printer object, you are prompted for a home directory and pairs that provide printer options. For example,

```
Spooled Printer: PRENSA
        Home Directory: local:>print-spooler>
        Other Options: zero or more pairs of a global name and a token
```

Hardcopy requests are stored in the home directory. The default for Symbolics computers is `local:>print-spooler>`.

Host Object Attribute: System-Type

System-Type

Specifies the operating system run on the host (as opposed to the hardware machine-type; see the section "The Machine Type Host Attribute"); a global-name (required). The Symbolics file system software and network software uses this information to communicate with a given host. Any operating system name is acceptable in this slot. The complete list of operating system names is available in the Network Information Center RFC1010. However, Genera only knows about some more common operating systems. The system types known to Genera are:

Value	*Software Type*	*Software Version*
lispm	Symbolics	Any
unix42	UNIX	4.2BSD and later
unix	UNIX	Prior to 4.2BSD
ultrix	Ultrix	Any
xenix	Xenix	Any
vms4.4	VMS	4.4 and later
vms4	VMS	4.0, 4.1, 4.2, and 4.3
vms	VMS	Prior to version 4
tops-20	TOPS-20	Any
tenex	TENEX	Any
its	ITS	Any
multics	MULTICS	Any
msdos	MS-DOS	Any
vm370	IBM VM	Any

If you provide an unknown type, the host object is created with type `random`; be sure to enter values correctly. For example:

```
System Type*: LISPM
```

Host Object Attribute: User-Property

User-Property

All objects contained within the namespace (hosts, sites, namespaces, printers, and users) are eligible to have a User-Property attribute. It consists of a pair whose first element is an indicator (like that of a property list) and whose second element is a token. The User-Property attribute holds any information that users choose to associate with an object. For example:

```
User-Property: ID-number 123-45-6789
```

Attributes for Objects of Class "User"

These attributes belong to objects of class "user" within a namespace. A user object can represent one of two things:

- A person.
- A daemon (pseudo-user).

Symbolics computers log in as daemon users when they act as file or mail servers, for example, or while they are performing maintenance functions.

In the namespace editor, required user object attributes have an asterisk by them.

User Object Attribute: Affiliation

Affiliation — Specifies the user's group affiliation; a single character. The character is arbitrary and can refer for example, to different sets of users.

```
Affiliation: Z
```

User Object Attribute: Birthday

Birthday — Specifies the user's birthday; a token.

```
Birthday: February 22
```

User Object Attribute: Home-Address

Home-Address — Specifies the user's home address; a token.

```
Home Address: Mount Vernon VA
```

The Home Host User Attribute

Home-Host — Specifies the host from which the user's lispm-init file is read (required). For example,

```
Home Host*: SHOOFLY
```

User Object Attribute: Home-Phone

Home-Phone — Specifies the user's home phone number; a token.

```
Home Phone: 202-999-1234
```

User Object Attribute: Lispm-Name

Lispm-Name — Specifies the name displayed in the status line; a token (required). For the Lispm-Finger service, the user name.

The Lisp variable **zl:user-id** is set from this attribute. It's usually similar to the actual name of the user object (but it uses upper- and lowercase letters). For example,

```
LispM Name*: GWash
```

User Object Attribute: Login-Name

Login-Name — Specifies the appropriate login name for each of several hosts; one or more pairs. The first element in each pair (a token) gives the login name. The second element provides the host object corresponding to that name.

In general, one Login-Name should exist for each host on which the user has an account. Login-Name is not a required attribute, but if it's not present, any server that requires that a Login-Name be presented to it will prompt for one (which might be inconvenient).

Passwords are not stored in the database. The Symbolics computer prompts the user for a password when one is required.

User Object Attribute: Mail-Address

Mail-Address — A pair that specifies the network mailbox at which the user will receive mail. The first element in the pair is the mailbox name (a token). The second element in the pair is a host. For example:

```
Mail Address*: Steve DREGS
```

More information is available about mailboxes. See the section "Symbolics Store-and-Forward Mailer".

User Object Attribute: Nickname

Nickname — Specifies a personal nickname; a token. Unlike host nicknames, user nicknames cannot be used to find a user object.

User Object Attribute: Personal-Name

Personal-Name — Specifies the user's personal name; a token (required). For example,

```
Personal Name*: Morse, Steve
```

User Object Attribute: Project

Project — Specifies what the user is working on; a token. For example,

```
Project: MacIvory
```

User Object Attribute: Pretty-Name

Pretty-Name — Most objects contained within the namespace (sites, other namespaces, networks, hosts, and printers) are eligible to have a Pretty-Name attribute. The Pretty-Name attribute specifies a name; a token.

The pretty name is the object name that appears on screen displays and in prompts. That is, an object's pretty name appears where people need to see it, while an object's actual name is used by software.

User Object Attribute: Remarks

Remarks — Specifies any information (like comments); a token. For example,

```
Remarks: "I cannot tell a lie."
```

User Object Attribute: Supervisor

Supervisor — Specifies for whom the user is working; a token. For example,

```
Supervisor: JO
```

User Object Attribute: Type

Type — Specifies a special user type such as "daemon". No value is needed for typical user objects.

User Object Attribute: Work-Address

Work-Address — A token that specifies a work (business) address. For example,

```
Work Address: The White House, Washington D.C.
```

User Object Attribute: Work-Phone

Work-Phone — A token that specifies the work (business) phone number. For example,

```
Work Phone: 202-555-1212
```

Attributes for Objects of Class "Printer"

These attributes belong to objects of class "printer" within a namespace. A printer object is a hardcopy output device.

In the namespace editor, required printer object attributes have an asterisk by them.

Printer Object Attribute: Body-Character-Style

Body-Character-Style

A list specifying the character style that should normally be used for this printer. The first element is the family; the second element is the face; the third element is the size. For example,

```
Body Character Style: SWISS.ROMAN.LARGE
```

See the section "Character Styles". If this attribute is not specified, the default character style is usually determined by the type of printer.

Printer Object Attribute: Character-Size

Character-Size — Specifies the size of a character in micas; width and height, in decimal. (A *mica* is 10 microns, or 1/2540 of an inch.)

Printer Object Attribute: Default-Font

Default-Font — Specifies the font that should normally be used for this printer; a token. If not specified, the default-font is usually determined by the type of printer.

Printer Object Attribute: DPLT-Logo

DPLT-Logo — This printer object attribute is no longer supported.

Printer Object Attribute: Fonts-Width-File

Fonts-Width-File — A token that specifies the name of the `fonts.widths` file for this printer. Use a full physical pathname instead of a logical pathname, like this

```
Font Widths File: A:>sys>stats>lgp-1>fonts.widths
```

Printer Object Attribute: Format

Format — Specifies the print formats supported by the device; a set of global-names (in addition to those implied by the Type printer attribute). For example,

```
Format: LGP
```

Common print formats include:

lgp
lgp2
lgp3
press
xgp
ascii
tektronix

The Header Font Printer Attribute

Header-Font — Specifies the name of the header font that should normally be used by this printer. If not supplied, the type of printer usually determines what this font will be.

Printer Object Attribute: Heading-Character-Style

Heading-Character-Style

A list that specifies the name of the character style that should be used for this printer. The first element is the family; the second element is the face; the third element is the size. For example,

```
Heading Character Style: SWISS.ROMAN.VERY-LARGE
```

See the section "Character Styles".

Printer Object Attribute: Host

Host — Specifies the host to which the printer is directly connected; a host object (required).

Printer Object Attribute: Interface-Options

Interface-Options — Specifies parameters of the hardware interface. Click on the correct values. For example,

```
Interface Options:
Unit: 1
Baud: 300  600  1200  1800  2000  2400  3600
4800  7200  9600  19200  56000
Other Options: zero or more pairs of a global name and a token
```

Printer Object Attribute: Interface

Interface — Specifies the type of interface by which this printer is attached to its host. Click on the correct choice. For example,

```
Interface: Serial  Elp  Other
```

Printer Object Attribute: Page-Size

Page-Size — A pair that specifies the size of the page in device units; width and height, in decimal. For example,

```
Page Size: 135 80
```

Printer Object Attribute: Pretty-Name

Pretty-Name — Most objects contained within the namespace (sites, other namespaces, networks, hosts, and printers) are eligible to have a Pretty-Name attribute. The Pretty-Name attribute specifies a name; a token.

The pretty name is the object name that appears on screen displays and in prompts. That is, an object's pretty name appears where people need to see it, while an object's actual name is used by software.

Printer Object Attribute: Printer-Location

Printer-Location — Describes the physical location of the printer; a list (three tokens). The first element identifies the building. The second element is the floor number. The third element is a textual description.

```
Printer Location: SCRC 3 Jennifer's office
```

Printer Object Attribute: Protocol

Protocol — Specifies protocols for direct (unspooled) printing; a set of global-names. If protocols are not specified, hardcopy service is invoked on the host to which the printer is directly connected.

Printer Object Attribute: Site

Site — The site where the printer is located; a site object. Generally all printers at a site are offered in menus of potential output devices for the destination of a hardcopy request.

Printer Object Attribute: User-Property

User-Property — All objects contained within the namespace (hosts, sites, namespaces, printers, and users) are eligible to have a User-Property attribute. It consists of a pair whose first element is an indicator (like that of a property list) and whose second element is a token. The User-Property attribute holds any information that users choose to associate with an object. For example:

```
User-Property: ID-number 123-45-6789
```

Attributes for Objects of Class "Network"

These attributes belong to objects of class "network" within a namespace. A network object is a computer network to which some hosts are attached.

In the namespace editor, required network object attributes have an asterisk by them.

Network Object Attribute: Global Network Name

Global-Network-Name

A token that, when specified at two sites, helps the Generic Network System to determine that networks at these sites are logically connected with one another.

More information is available about Global Network Names. See the section "Sync-Link Gateways".

Network Object Attribute: Nickname

Nickname — Specifies alternate names for the network; a set of names by which the network may be found.

Network Object Attribute: Site

Site — Specifies the site at which this network is located; a site object.

```
Site: HARVARD
```

Network Object Attribute: Subnet

Subnet — Specifies the characteristics of a network's subnetwork. The first element is a token that names the subnet. The second element is one or more pairs of global-names and tokens that provide extra information about the subnet. For example,

```
Subnet: 81 cable-start "Room 2" cable-end "Room 15"
```

More information is available about the how the Subnet attribute is used. See the section "Symbolics Dialnet".

Network Object Attribute: Type

Type — A global-name that specifies a network type. For example,

```
Type*: INTERNET
```

Common network types include:

CHAOS — A network using the Chaos protocols. Addresses are 16-bit numbers represented in octal. For example,

```
17006
```

INTERNET — A network using the DD Internet protocols. Addresses are the 32-bit Internet addresses as four octets, represented in decimal, separated by periods. For example,

```
10.0.0.6
```

DNA — A network using the DECnet Digital Network Architecture protocols. DNA addresses are 16-bit quantities, where the high-order 6 bits constitute the area, and the low-order 10 bits constitute the node number; they are expressed in decimal notation. For example,

```
3.7
```

DIAL — A direct-dial telephone network. Usually there is only one of these, called "dial" by convention. Addresses are telephone numbers governed by the dialing conventions of the installation. For example,

```
15551212
```

X25 — A packet-switching network with a CCITT Recommendation X.25 interface. Addresses are X.121 addresses. For example,

```
311061700138
```

GATEWAY-PSEUDONET — A network actually implemented by direct connection of a gateway to a terminal line. Address is *service-name = contact name* on gateway host. For example,

```
tty-login=prime
```

Network Object Attribute: User-Property

User-Property — All objects contained within the namespace (hosts, sites, namespaces, printers, and users) are eligible to have a User-Property attribute. It consists of a pair whose first element is an indicator (like that of a property list) and whose second ele-

ment is a token. The User-Property attribute holds any information that users choose to associate with an object. For example:

```
User-Property: ID-number 123-45-6789
```

Attributes for Objects of Type "File System"

Host

Specifies the host that the file system resides on; a host object (required).

```
Host: MARS
```

Type

Must always be DBFS (required). Other values are reserved for future expansion.

```
Type: DBFS
```

Root Directory

Specifies a pathname of a FEPFS file. That file contains the directory of the Statice File System. The pathname should always start with FEP*n*: and end with the .UFD file extension.

```
Root Directory: FEP1:>Iris.UFD
```

Pretty Name

Specifies a name for the file-system to use when showing the name; a token (required).

```
Pretty Name: Iris
```

Nickname

Specifies alternate names for the network; a set of names. The file system may be found by these names.

```
Nickname: IRE
```

Short Name

Specifies additional nicknames; a set of names. A short-name is used when a program wants to display a host's name without using up too much space. A short-name is used for both input and output. This is also used in the printed representation of pathnames.

```
Short Name: I
```

User Property

User-Property

All objects contained within the namespace (hosts, sites, namespaces, printers, and users) are eligible to have a User-Property attribute. It consists of a pair whose first element is an indicator (like that of a property list) and whose second element is a token. The User-Property attribute holds any information that users choose to associate with an object. For example:

```
User-Property: ID-number 123-45-6789
```

Statice automatically places several user properties into file-system objects. The User Properties named PARTITION have values that are pathnames of the FEP files which are the partitions that make up the file system. The database is stored in a number of partitions. For more information on partitions: See the section "Create Statice File System Command".

The User Property named LOG-DESCRIPTOR-FILE-ID is the unique ID of Statice's "log descriptor" file, an internal file used to store various per-file-system information.

The User Property named DBFS-DIR-ROOT-FILE-ID contains, in the form of a string, the internal unique ID of the special database in the file system that stores the hierarchical directory structure of the file system. This is established when the Statice File System is created, and you should never change it.

Lisp Functions for Namespace Database Administration

Under normal circumstances, Namespace Editor and CP commands are sufficient for performing administrative tasks within the namespace database. Occasionally, however, the Lisp functions described within this section can be useful.

neti:read-object-file-and-update *namespace class-name* *Function*

Updates the namespace database from an object file. *namespace* can be a namespace object or the name of one. This function is used for namespaces which are maintained outside of the Symbolics namespace database, but which should be accessible to it. It reads an object file (usually generated from some external source of information) and makes the namespace database agree with it by adding, changing, and deleting objects. The changes and log files are updated. It can be invoked only on the primary namespace server for the namespace to be updated.

```
(neti:read-object-file-and-update
   :arpanet :host)
```

Note that this function does not update the object file itself; you should do this in Zmacs. A typical use of **neti:read-object-file-and-update** is as follows:

1. Disable Services on the primary namespace server.

2. Edit the object file (such as YOURSITE-HOSTS.TXT) with Zmacs.

3. Do `(neti:read-object-file-and-update "SCRC" :host)`

4. Enable Services on the primary namespace server.

neti:prune-namespace-changes-file *namespace starting-timestamp* *Function*

Eliminates the record of changes to *namespace* before *starting-timestamp*. This reduces the amount of information which must be processed by the primary namespace server when it is booted. The changes file is best pruned only when there are no world load files that were saved before the earliest remaining change; they will take quite awhile to boot.

Making, Distributing, and Using Worlds

Symbolics distributes new Genera software on tape. We distribute world loads, called distribution worlds, along with documentation sources, examples, fonts, and non-loaded systems. You can customize a distribution world for your site's needs. The sequence of steps by which you do this will vary. This section outlines the procedures for customizing and saving worlds for your site. All the commands you'll need to do this are documented in this section.

If you want to load a new world and save an incremental version of it, here is the basic sequence of steps you might take:

1. Boot the new world. For information on booting: See the section "Booting a World".

2. Use the Set Site command to make the world site-specific.

3. Use the Optimize World command to optimize paging performance. To do this from a program, use the function **(si:reorder memory)**.

4. Use the Save World command to save an incremental world on your machine.

If you want to add some special programs, or systems, to the initial distribution world load, and then create an incremental world, this is the sequence of steps you might take:

1. Boot the new world. For information on booting: See the section "Booting a World".

2. Use the Set Site command to make the world site-specific.

3. Use the Load System command to load special software into your world.

4. Use the Optimize World command to optimize the world.

5. Use the Save World Incremental command to save an incremental world on your machine.

If you want to add patches (updates to the software distributed by Symbolics) to the initial distribution world load, and then create an incremental world, this is the sequence of steps you might take:

1. Boot the new world. For information on booting: See the section "Booting a World".

2. Use the Set Site command to make the world site-specific.

3. Use the Load Patches command to add updated software to your world.

4. Use the Optimize World command to optimize paging performance. To do this from a program, use the function **(si:reorder memory)**.

5. Use the Save World command to save an incremental world on your machine.

For information about Incremental Disk Save (IDS): See the section "Using the Incremental Disk Save (IDS) Facility".

For information about the Optimize World command: See the section "Optimizing Worlds".

Note: If you want to supply a world to a Symbolics 3600-family machine that does not have a FEP EPROM version 127 or greater, you should follow one of the procedures listed above, but use the Save World command with the :Complete, instead of :Incremental, argument.

If your site needs to build a distribution world, follow these steps:

1. Boot the new world. For information on booting: See the section "Booting a World".

2. Use the Set Site command to make the world site-specific.

3. Use the Load System command to load any additional software.

4. Use the Set Site command with the site name as Distribution.

5. Use the function **(si:full-gc)** to garbage-collect the world.

6. Use the Optimize World command to optimize paging performance. To do this from a program, use the function **(si:reorder memory)**.

7. Use the Save World command to save the complete world.

Using the Incremental Disk Save (IDS) Facility

The Incremental Disk Save (IDS) facility allows you to save modified worlds. IDS saves a world (called the *incremental world*) by copying only those pages of an ancestor (or *parent world*) that have changed; incremental world loads require less disk space than complete world loads do.

You can make multiple (different) incremental worlds from one parent world, and save each with a minimum of disk space. IDS-worlds run slightly slower than non-IDS worlds because they utilize extra wired memory.

Note: Keep all of the ancestors for any incremental worlds you intend to use; descendents require blocks from their ancestors. If you have a Symbolics 3600-family machine, and you are netbooting an incremental world, you may keep the ancestor(s) on the netboot server (rather than on the local host).

To perform an incremental disk save, boot an existing world with IDS enabled. See the section "Enable IDS FEP Command". After making site-specific modifications to the world (by loading private patches, and any systems that its users will need), save the world by using the Save World command, specifying the type of world (incremental) that you want to save. See the section "Save World Command".

Note: Do not use **gc-immediately** or **si:full-gc** on a world prior to using IDS. Instead, Symbolics recommends that you use the ephemeral-object garbage collector (EGC). By default, EGC is enabled.

For information about the command that shows the IDS parent worlds for a specified world load file: See the section "Show IDS Parents Command".

For information about the command that shows the IDS children for a specified world load file: See the section "Show IDS Children Command".

For information about the command that shows the IDS files for a specified host: See the section "Show IDS Files Command".

Optimizing Worlds

If you load special software or programs into distribution worlds, use the Optimize World command to improve the new worlds' paging performance. (To do this from a program, use the function **(si:reorder-memory)**.) If you load the distribution world and then customize it for your site without loading any additional programs, you needn't use the Optimize World command.

Paging performance measures the speed at which accesses occur from virtual memory. Optimization software moves related functions and data so that they reside in contiguous (virtual) memory locations. After a world has been optimized, it runs with improved paging performance.

The optimization software does not move objects that were originally part of the distribution world; this world is optimized before you receive it. The software moves only new objects that you load into the distribution world.

Use the Optimize World Command after loading any site-specific software or layered products, and before saving the world.

Note: If you optimize an IDS world whose parent world was not (but should have been) optimized, your IDS world will be significantly larger than neccessary.

Use the Optimize World command on IDS worlds only if:

- The parent world was optimized, or
- The parent world didn't need optimization.

If you follow this rule, optimization will not increase the size of your IDS world.

Since distribution worlds are optimized before leaving Symbolics, if you site-configure a distribution world (but do not load any systems) you need not optimize it again. (This saves you some time in your world-building process.)

Direct Calls: a Linking Feature for Ivory-based Machines

The Ivory architecture provides Direct Calls, a fast mechanism for function calls that is mostly usable for benchmarking and application delivery.

In a normal Lisp call (an "indirect" call), the caller function has a pointer to the *function cell* containing the function to be called. When the call instruction is executed, it fetches the callee function from the function cell, and starts execution at the entry instruction of that function. The entry instruction sequence checks that the proper number of arguments was passed, initializes optional and keyword arguments, and then proceeds to execute the body of the called function.

The normal call is called "indirect" because it fetches the contents of a function cell (indirects through it) rather than addressing the callee function directly. Lisp implementations typically implement calls as indirect calls in order to efficiently support redefinition at runtime: When a function is redefined, all the Lisp system has to do is change the contents of the function cell, and all callers will immediately address the new definition.

In a direct call, the caller addresses the callee function directly, without going through a function cell. For Lisp systems that implement function calls using the direct method, redefinition must change *every* caller of a function to address the new definition. This is typically very slow.

Another optimization is possible when calls are implemented directly. Since relatively simple static analysis can determine how many arguments are being passed to a function, a direct call can often skip the preamble instructions that check for the proper number of arguments and initialize optional arguments.

Genera 8.0 provides a linker for Ivory-based machines that performs both of the above optimizations. Depending on the application, its use can result in substantial performance improvements. The linker is not fully integrated with Genera. If there are direct calls to a function, and there is an attempt to redefine it, an error is signaled. Proceed options allow you to unlink definitions to a function before redefining, or to proceed without unlinking.

To globally link all functions, use `(cli::link-to-functions t)`. To globally unlink them, use `(cli::unlink-to-functions t)`. If you need finer control of which existing functions should be linked or unlinked, refer to **cli::link-to-functions** and **cli::unlink-to-functions** for further information.

Regardless of whether any functions are linked or not, newly compiled or loaded functions are always unlinked.

Note: Because of architectural limitations, linking does not work on 3600-family machines, In order to get the additional performance benefit of linking, you must use an Ivory-based processor.

cli::link-to-functions *functions* &optional *link-noter verbose* *Function*

Links all calls to the functions specified by *functions*. *functions* is either a list of functions and/or function specs, or the symbol **t**, meaning all functions. This process takes up to twenty minutes, depending on your system configuration and the amount of software loaded.

cli::unlink-to-functions *functions* &optional *unlink-noter verbose* *Function*

Unlinks all calls to the functions specified by *functions*. *functions* is either a list of functions and/or function specs, or the symbol **t**, meaning all functions. This process takes about five minutes, depending on your system configuration and the amount of software loaded.

Enabling the Who-Calls Database

The **who-calls** database helps to locate the callers of variables, functions, or macros.

The **who-calls** database is a cache that maps *names* (which are symbols) to code and variables that use the symbols in some way. A name can be used as a con-

stant, a variable, a function, a macro, an instance variable, or a condition, for example.

By default, the Set Site command automatically calls the function **(si:enable-who-calls :new)**. This enables the **who-calls** database to record the callers in any layered products, special software, or programs loaded into the world (after the site has been set).

More information is available on **si:enable-who-calls** and related functions. For more information, see the section "Lisp Functions Related to the Who-Calls Database".

Compressing the Who-Calls Database

After you use the function **si:enable-who-calls** with the argument best suited for the type of database you want to create, you can compress the database by using either **(si:compress-who-calls-database)** or **(si:full-gc)**.

It is best to use **(si:compress-who-calls-database)**, since it is faster and does not preclude your using Incremental Disk Save (IDS). (Using Incremental Disk Save (IDS) after **(si:full-gc)** renders your world the same size as the world from which you started.)

If you want to have the entire body of Symbolics-supplied software in your **who-calls** database, there are different modes in which you can use the function **si:enable-who-calls** during the customization of the distribution world.

Here are examples of the ways in which you can couple **si:enable-who-calls** in different modes with **(si:full-gc)** and **(si:compress-who-calls-database)**:

1. Use the form **(si:enable-who-calls ':all-no-make)** followed by the form **(si:full-gc)**.

   (si:enable-who-calls ':all-no-make) creates a callers database that includes only new functions. When you request **si:full-gc** the entire database is created. This takes a long time and about 2000 pages of storage.

2. Alternatively, use the form: **(si:enable-who-calls ':all)** followed by the form **(si:compress-who-calls-database)**.

   (si:enable-who-calls ':all) creates a full callers database. This also takes a long time and about 2000 pages of storage. **si:compress-who-calls-database** compresses the who-calls database by garbage-collecting the database.

3. Use the form **(si:enable-who-calls ':explicit)** followed by either **(si:compress-who-calls-database)** or **(si:full-gc)**.

   (si:enable-who-calls :explicit) enables you to add items to the callers database explicitly, by using **(si:add-files-to-who-calls-database)** or **(si:add-system-to-who-calls-database)**.

Note: If you use **(si:enable-who-calls :explicit)** or **(si:enable-who-calls :new)**, load only a small amount of software into the world, and then save the world, there is no advantage to compressing or doing a full garbage collection.

Lisp Functions Related to the Who-Calls Database

This section contains information about **si:enable-who-calls** and functions that you will want to use with it.

si:enable-who-calls &optional *mode* *Function*

mode describes how the who-calls database should record the callers of any function. For more information about the **who-calls** database, see the section "Enabling the Who-Calls Database".

:all If you want to include callers of the Symbolics-supplied software (that is, software contained in the distribution world) in the database, use **:all**. This enables you to create the database once and then save it when you save the world. (When used with this argument, **si:full-gc** would discard the existing database and then remake it).

:all-remake Includes callers of the Symbolics-supplied and site-specific software in the database. Use this if you do not want to perform a **si:full-gc**. (When used with this argument, **si:full-gc** would discard the existing database and then remake it).

:new Enables the **who-calls** database to record the callers in any layered products, special software, or programs loaded into the world (after the site has been set). The Set Site command uses this argument by default. **:new** does not cause the callers of software in the distribution world to be recorded.

:all-no-make Enables the **who-calls** database to record the callers in any layered products, special software, or programs loaded into the world (after the site has been set), and does not cause the callers of software in the distribution world to be recorded until **si:full-gc** is performed. Once **si:full-gc** is performed, those callers (for software in the distribution world) are recorded.

:explicit If you want only explicitly-named files to be in the database, use the function **si:enable-who-calls** with the argument **:explicit**.

Note: Creating a full database takes a long time and about 2000 pages of storage.

si:compress-who-calls-database *Function*

Makes the **who-calls** database more compact and efficient. Call this function after **si:enable-who-calls**. With the function **(si:enable-who-calls ':all)**, the function **si:compress-who-calls-database** takes a long time to complete its job. However, it is faster than using **si:full-gc**, and you can perform an Incremental Disk Save (IDS) afterwards. See the section "Using the Incremental Disk Save (IDS) Facility".

si:full-gc &key *system-release* *Function*

Garbage-collects the entire Genera virtual memory environment, including some static areas. However, because static areas change slowly and are not likely to contain much garbage, use **gc-immediately** or the command Start GC :Immediately instead. See the section "Start GC Command". **si:full-gc** leaves the garbage-collector facilities in the state that it originally finds them, that is, with the same dynamic and ephemeral option settings.

If you use **si:full-gc**, call it with no arguments. The option **:system-release** is reserved for use by Symbolics. **si:full-gc** does an immediate, complete, nonincremental garbage collection as a preparation for immediately saving a world.

si:full-gc performs these operations:

- Resets temporary areas.
- Sets up the static areas to be cleaned up.
- Flips.
- Scavenges and flushes oldspace.
- Makes static areas static again.

It is not useful to perform an Incremental Disk Save (IDS) after running **si:full-gc**. Perform a complete disk save, instead.

Note: The Command Processor command Optimize World is the preferred high-level interface to the functions **si:full-gc**, **si:reorder-memory**, and **si:optimize-compiled-functions**. See the section "Optimize World Command".

Using the Initialization Lists invoked by si:full-gc

Two initialization lists, accessed through the **full-gc** and **after-full-gc** keywords to **add-initialization**, are run by **si:full-gc**. See the section "Introduction to Initializations".

si:full-gc runs the forms on the **full-gc** initialization list and then garbage-collects without multiprocessing (inside a **without-interrupts** form). The machine essentially "freezes" and does nothing but garbage collection for the duration. This operation takes 20 minutes or more, depending on the size of the world. After the garbage collection is completed, and before it reenables scheduling and returns, **si:full-gc** runs the forms on the **after-full-gc** initialization list.

full-gc is a system initialization list. You can add forms to it by using the **:full-gc** keyword in the list of keywords that is the third argument of **add-initialization**. The **full-gc** initialization list is run just before a full garbage collection is performed by **si:full-gc**. All forms are executed without multiprocessing, so the evaluation of these forms must not require any use of multiprocessing: they should not go to sleep or do input/output operations that might wait for something.

Typical forms on this initialization list reset the temporary area of subsystems and make sure that what is logically garbage has no more pointers to it.

Creating a World-Build Script File

Here is a sample world-build script file. It contains some forms that you might put into a world-build script file for your site.

Note: This file is one that contains forms enclosed in a wrapper function; it isn't a script file in the traditional sense.

Read the comments (prefaced by three semi-colons) for an explanation of the file's contents.

```
;;; -*- Syntax: common-lisp; Base: 10; Mode: LISP; Package: SYSTEM-INTERNALS; -*-

;;; a large function to do the entire job of building
;;; a world, given that you execute it in a site-configured environment

;;; use this to create a user or server world
(defun produce-world (&key server)

    (format t "~2& Enabling the EGC")

    ;;; make sure that the GC is in a consistent condition
    (gc-on :ephemeral t :dynamic nil)

    ;;; disable screen-dimmer to avoid interrupts-off surprises
  (let ((tv:*dim-screen-after-n-minutes-idle* nil))
    (setq time-start (time:get-universal-time)
    ;;; disable the services, to avoid any unwanted network interactions
    (disable-services)
    ;;; turn the global more breaks off, in case not already done
    (setq tv:more-processing-global-enable nil)
    ;;; this assures that someone is logged in to the machine
    (fs:force-user-to-login)

    (format t "~2& *** Constructing your site's world, System ~D.~2%~
       ~4TServices disabled, more processing on locally, off globally,
       logged in as ~A.~@@ ~4TLoading patches." (get-system-version) user-id)
```

```
    ;;; load patches to make sure you're up to the current patch level
    (load-patches nil :query nil)

    (format t "~2& *** Up to current patch level; loading added systems.~%")

    ;;; now load the extra systems that you want in your world
    (flet ((load-a-system (name)
             (unless (sct:find-system-named name nil t t)
               (format t "~2&    Loading ~A.~2%" name)
               (sct:load-system name :query :no-confirm))))
          (load-a-system "metering")
          (when server
            (load-a-system "Print")
            (load-a-system "Mailer"))



    (format t "~2& *** Added systems loaded.~%")

     ;;; compile the who-calls database
    (si:enable-who-calls :all-no-make)

    ;;; now do a full gc.  This takes about 1.75 hours
    (format t "~2% *** Beginning Full-GC.")
    (full-gc)

      (format t "~2% *** Beginning Reorder-Memory.")
        (reorder-memory :incremental nil
                        :run-without-interrupts t) ;;;run-without-interrupts is faster

    ;;; Last (and least) make this function disappear
    (fundefine 'produce-world)

    ;;; Print final statistics.
    (format t
            "~2& *** Full-GC, Reorder-Memory and final parameter settings complete.

             If everything~@@ ~5@@Tlooks OK, please save the result via Save World .~2%")
    (values)))


(format t "~2&Start the world production by calling ")
(present '(produce-world) 'sys:form)
(format t ".~%")
```

Creating a Site System for Holding Private Patches

If you have a software service contract, you might receive private patches as workarounds for problems you report. These small pieces of code (usually modifications to functions from system sources) should be kept separate from the system for which they are patches, because their patch number might cconflict with official patches for that system that you might get later in a Software ECO distribution. The right place to keep them and to patch them is in a *site system*.

A site system is a small system that contains software you want loaded by all the Symbolics machines at your site. It is defined like any other system (see the section "Defining a System"). For example, if your site were named ACME, you might define a site system to hold a patch and some local hack like this:

```
(defsystem ACME-SITE
    (:pretty-name "Acme Site System"
     :default-pathname "SYS:IN-HOUSE;"
     :patchable t
     :advertised-in (:herald :finger :disk-label)
     :initial-status :released
     :before-patches-initializations (sct:set-component-systems-advertised-in nil)
     :maintaining-sites :acme
     )
  (:module trap-handler "trap-handler-patch")
  (:serial
      "trap-handler-patch"
      "finger-hack"
    ))
```

To register this system as a site system, you then add the `Site-System` attribute to your Site object in the Namespace. The object `Site ACME` in ACME's namespace would be given the attribute:

```
Site-System: ACME-SITE
```

A site system becomes a collection of miscellaneous pieces and modifications to standard systems. There are some guidelines to observe when adding things to it:

1. If you are adding or modifying something that is expected to have an indefinite lifetime in the system, place this code into its own source file in system source directory (SYS:IN-HOUSE; in our example) with an appropriate package, and add the source to the list of sources in the system declaration file. Then patch the change for the purposes of the current system.

2. If you are adding a workaround from software support that will have a lifetime of only the current version of Genera, place the file in the system directory but do not include it in the system declaration. Just patch the code into the site system, and clearly mark it as a temporary patch in the patch comment.

3. If something was installed provisionally in this site system, and it is superseded by an ECO from Symbolics, you must revoke it from your site system so that it does not shadow the official fix:

   a. If it exists in a source in the system declaration, remove that source file name from the list in the system declaration. If appropriate, make a patch to undo the removed code in the current world.

   b. If it exists in a patch, revoke the patch:

      i. Use m-X Resume Patch for that patch number.

      ii. Delete all the code from the patch file.

      iii. Do m-X Finish Patch and change the patch comment to say something like "Superseded by Symbolics ECO".

   c. Both of the above may apply.

4. Your site system should be recompiled every time you receive a new release from Symbolics.

***Caveat**: When you receive ECO or new releases, previous patches and workarounds might be superseded by a change to the appropriate system, and you would then have loading order clashes. You must exercise caution to recognize this situation as ECOs are received, and withdraw the site-system patch.*

It is advisable to remove all Symbolics-provided private patches in your site system when you receive a new release, until you can verify that they are still needed.

World-Related Commands and Functions

These commands and functions are often useful to site maintainers who make, distribute, and use worlds. For your convenience, we have arranged the commands in alphabetical order; the commands appear first.

Copy World **Command**

Copy World *file destination keywords*

Makes a copy of *file* (by default, a world load). This includes the specified world as well as any Incremental Disk Save (IDS) worlds on which it was built. See the section "Using the Incremental Disk Save (IDS) Facility". Copy World works from remote terminals. Copy World can also be used to copy netboot cores. You can boot a world from a remote world server with only a netboot core on your FEP. See the section "Netbooting".

file A FEP file specification; the world to copy. The default is constructed from the version of the world that you have booted.

destination A FEP file specification; the pathname for the new world. The default is a wildcard pathname assuring the correct hierarchical pathname relationship for the parent world and an IDS world.

Note: The `.ilod` file extension indicates world-load files for Ivory-based machines, just as the `.load` file extension indicates world-load files for Symbolics 3600-family machines. Files with the `.ilod` extension can be copied only between Ivory-based machines. Files with the `.load` extension can be copied only between Symbolics 3600-series machines.

After you issue the Copy World Command, Genera puts up a menu allowing you to specify the actions you want it to take:

```
There are 40450 blocks total in the files to be transferred.
  There are 809 blocks free on FEP1.
  You need 39641 more blocks.
  Possible actions to make space right now: Run dired  Expunge directory  Selectively delete

Parent IDS files to transfer: All parents  Missing parents  Just requested files  Selective
Update boot file to load FEP1:>Base-System-372-0.load.1: Yes  No
  Boot file to update: FEP0:>boot.boot.newest
  Update FEP0:>boot.boot.newest to load microcode 410: Yes  No
Transfer mode: Transfer-And-Checksum  Transfer-Only  Checksum-Only
Attempt automatic error recovery: Yes  No
<ABORT> aborts, <END> uses these values
```

Figure 105. Copy World

keywords :Automatic, :End Block, :File Set, :More Processing, :Output Destination, :Query, :Start Block, :Transfer Mode, :Update Boot File.

:Automatic {Yes, No} Whether or not to attempt automatic error recovery. The default is Yes.

:End Block {*integer*} The number of the last block to copy from source. The default is the last block, meaning copy until the end.

:File Set {All parents, Missing parents, Just Requested Files, Selective} Which parent IDS files to transfer. The default is Missing parents.

:More Processing {Default, Yes, No} Controls whether **More** processing at end of page is enabled during the output to interactive streams. The default is Default.

If No, output from this command is not subject to **More** processing.

If Default, output from this command is subject to the prevailing setting of **More** processing for the window. (See the section "FUNCTION M".)

If Yes, output from this command is subject to **More** processing unless it was disabled globally (see the section "FUNCTION M").

:Output Destination
{Buffer, File, Kill Ring, Printer, Stream, Window} Where to redirect the typeout done by this command. The default is the stream ***standard-output***.

:Query {Yes, No} Whether or not to present a menu of transfer parameters. The default is Yes.

:Show Blocks Copied
{Yes, No} Whether to print block numbers for blocks finished copying, every 100 blocks. The default is No, the mentioned default is Yes.

:Start Block {*integer*} The number of the block to start copying from the source. The default is 0, meaning begin at the beginning.

:Transfer Mode {Transfer-and-Checksum, Transfer-Only, Checksum-Only} Whether to verify the integrity of the copied world. The default is Transfer-and-Checksum. You can use Checksum-Only to checksum a band that you copied previously but were unable to checksum due to network problems.

:Update Boot file {*FEP-file-spec*, none}. Boot file to update to load the new world. The default boot file for IDS or complete worlds is boot.boot. The default for netboot cores is none.

Define Site **Command**

Define Site *site-name*

Defines a new site.

Type the Define Site command immediately after you boot a new distribution world when you want to define a new site and namespace; it brings up a menu to create a new namespace called *site-name*; when you start this dialogue the local host is, by default, the site's:

- Primary namespace server.
- SYS host.
- Host for storing namespace database files.
- Host for bug reports.

If you want non-local host(s) to perform any of these jobs, provide their primary network addresses and operating system types in the appropriate menu slots.

During the Define Site dialogue, the namespace database files (object files, log files, changes files, and a descriptor file) are created for you on the file system of the machine you specify as the namespace server. Make sure the file system exists on a host accessible to the namespace server machine before you issue the Define Site command. For more information about the namespace database files, see the section "Namespace Database Files".

The namespace server for the new site will have these initial default attributes when the Define Site command is used (*nnnnn* represents a valid octal Chaos address):

```
System Type*: LISPM
Machine Type: 3600
Service: CHAOS-STATUS CHAOS-SIMPLE CHAOS-STATUS
Service: SHOW-USERS CHAOS NAME
Service: TIME CHAOS-SIMPLE TIME-SIMPLE
Service: UPTIME CHAOS-SIMPLE UPTIME-SIMPLE
Service: LOGIN CHAOS TELNET
Service: LOGIN CHAOS SUPDUP
Service: LOGIN CHAOS 3600-LOGIN
Service: SEND CHAOS CONVERSE
Service: SEND CHAOS SEND
Service: NAMESPACE CHAOS NAMESPACE
Service: NAMESPACE-TIMESTAMP CHAOS-SIMPLE NAMESPACE-TIMESTAMP
Service: LISPM-FINGER CHAOS-SIMPLE LISPM-FINGER
Service: FILE CHAOS NFILE
Service: FILE CHAOS QFILE
Service: CONFIGURATION CHAOS CONFIGURATION
Address: CHAOS 12345
```

See the section "Define Site Dialogue".

Load System **Command**

Load System *system keywords*

Loads a system into the current world.

system — Name of the system to load. The default is the last system loaded.

keywords — :Component Version, :Condition, :Include Components :Load Patches, :More Processing, :Output Destination, :Query, :Redefinitions Ok, :Silent, :Simulate, :Version

:Component Version
{Released, Latest, Newest, *version-designator*} The version of any component systems to load. Released means the version designated as released in the journal file. Latest means the most recent version recorded in the journal file. Newest means

to ignore the versions in the journal file and just find the newest files. The default is the version with which the system was compiled.

:Condition {Always, Never, Newly-Compiled} Under what conditions to load each file in the system. Always means load each file. Newly-compiled means load a file only if it has been compiled since the last load. The default is Newly-Compiled.

:Include Components
{Yes, No} Whether to load component systems. The default is Yes.

:Load Patches {Yes, No} Whether to load patches after loading the system. The default is Yes.

:More Processing {Default, Yes, No} Controls whether **More** processing at end of page is enabled during the output to interactive streams. The default is Default.

If No, output from this command is not subject to **More** processing.

If Default, output from this command is subject to the prevailing setting of **More** processing for the window. (See the section "FUNCTION M".)

If Yes, output from this command is subject to **More** processing unless it was disabled globally (see the section "FUNCTION M").

:Output Destination
{Buffer, File, Kill Ring, Printer, Stream, Window} Where to redirect the typeout done by this command. The default is the stream ***standard-output***.

:Query {Everything, Confirm-only, No} Whether to query before loading. Everything means query before loading each file. Confirm-only means create a list of all the files to be loaded and then ask for confirmation before proceeding. No means just go ahead and load the system without asking any questions. The default is No. The mentioned default is Everything.

:Redefinitions Ok {Yes, No} Controls what happens if the system asks for confirmation of any redefinition warnings during the loading process. Yes means assume that all requests for confirmation are answered yes and proceed. No means pause at each redefinition and await confirmation. The default is No. The mentioned default is Yes. This allows you to start loading a system that you know will take a long time to load and leave it to finish by itself without interruption for questions such as "Warning: *function-name* being redefined, ok? (Y or N)".

:Silent {Yes, No} Whether to turn off output to the console while the system is loading. The default is No. The mentioned default is Yes.

:Simulate {Yes, No} Print a simulation of what compiling and loading would do. The default is No. The mentioned default is Yes.

:Version {Released, Latest, Newest, *version-designator*} Which version number to load. Released means the version designated as released in the journal file. Latest means the most recent version recorded in the journal file. Newest means to ignore the versions in the journal file and just find the newest files. The default is Released.

Note: This command only loads a system. If you want to compile and load a system, see the section "Compile System Command".

Load Patches **Command**

Load Patches *system keywords*

Loads patches into the current world for all systems, locally maintained systems, or the indicated systems.

system {All Local *system-name1, system-name2* ... } The system(s) for which to load patches. The default is All.

keywords :Dangerous Patch Action, :Excluding, :Include Components, :More Processing, :Output Destination, :Query, :Save, :Show

:Dangerous Patch Action
{Skip, Query, Load} Whether to skip loading *dangerous* patches, that is, patches that might make data structures in your world inconsistent, causing unexpected behavior. The default is Skip.

:Excluding {*System(s)*} Excludes loading patches for these systems.

:Include Components
{Yes, No} Whether to load patches for any component systems. The default is No. The mentioned default is Yes.

:More Processing {Default, Yes, No} Controls whether **More** processing at end of page is enabled during the output to interactive streams. The default is Default.

If No, output from this command is not subject to **More** processing.

If Default, output from this command is subject to the prevailing setting of **More** processing for the window. (See the section "FUNCTION M".)

If Yes, output from this command is subject to **More** processing unless it was disabled globally (see the section "FUNCTION M").

:Output Destination
{Buffer, File, Kill Ring, Printer, Stream, Window} Where to redirect the typeout done by this command. The default is the stream ***standard-output***.

:Query {Yes, No, Ask} Yes asks for confirmation before beginning the load patches process and again before loading each patch. Ask asks whether or not it should query before each patch, and then for confirmation before beginning the load patches process. The default is No. The mentioned default is Yes.

:Save {*pathname*, Prompt, No-Save} The file in which to save the world with all patches loaded. Omitting this keyword means do not save the world. The mentioned default is Prompt, which means save the world and then prompt for a pathname.

:Show {Yes, No, Ask} Whether to print the patch comments as each patch is loaded. The default is Yes.

See the function **load-patches**.

Optimize World Command

Optimize World *keywords*

Optimizes the world that is currently loaded into your enviroment by reorganizing the world to improve paging performance. Use this command after you load site-specific software or layered products into a distribution world, and before you save it.

Note: If you optimize an IDS world whose parent world was not (but should have been) optimized, your IDS world will be significantly larger than neccessary.

Use the Optimize World command on IDS worlds only if:

- The parent world was optimized, or
- The parent world didn't need optimization.

If you follow this rule, optimization will not increase the size of your IDS world.

Since distribution worlds are optimized before leaving Symbolics, if you site-configure a distribution world (but do not load any systems) you need not optimize it again. (This saves you some time in your world-building process.)

When you enter the Optimize World command, you are prompted for confirmation. Once the program has finished, a message appears. Optimization typically takes about one-half hour to execute, but this time period can vary according to the size

of the world load and the total amount of main memory that's available when you execute the command.

During the time that Optimize World is running your machine does not respond to the network or keyboard; you cannot use your machine while optimization is in process.

keywords :Show

:Show Displays the progress of the optimization process on the screen.

Save World **Command**

Save World (Complete or Incremental) *pathname*

Saves the current world. The system prompts for (Complete or Incremental) if Incremental Disk Save is enabled. Specify Complete to save the entire world, or Incremental (if enabled) to perform an Incremental Disk Save. The default is Complete.

pathname The pathname for the saved world. The default is the FEP file specification for the local machine, based on the version number of the current system and on whether this is a complete or incremental save.

A complete save yields a pathname of Genera-*major-minor* or System-*nnn-mmm*. An incremental save changes this default in the following way:

- "Genera-" is prefaced with "Inc-"
- "System-" is replaced by "Inc-".
- "-from-" is appended to the name.
- A shortened version of the loaded world name (the pathname that appears in the first line of the herald) is appended to the name.
- If the result is longer than 32 characters (the limit of filenames in the FEP file system), the filename is truncated and the last 4 characters within the limit are replaced with "-etc".

More information is available about saving incremental worlds. See the section "Using the Incremental Disk Save (IDS) Facility".

Set Site **Command**

Set Site *site-name*

Configures the local distribution world to be an already existing site.

site-name {*name*, get-from-network} The name of your site.

Any further arguments are entered through an AVV menu that adjusts depending on the parameters needed. The Set Site command also fully supports multiple sites within one namespace so the site name does not have to match the namespace name, although one site in any namespace must have a name that is the same as the namespace name.

If the Set Site command is used when the local machine is already registered in a site, the current site is changed to the distribution site before changing over to the new site. This is to eliminate any problems with dangling references to the previous site.

The Set Site command enables your machine to identify all objects included in the site's namespace database. The namespace database for each site is stored in the file system accessible from a machine called the namespace server.

In order for the Set Site command to work, your machine must be a registered host in the site's namespace. If the site's namespace server is not the local host, you must know the namespace server's name and network address.

See the section "Set Site Dialogue".

Show FEP Directory **Command**

Show FEP Directory *keywords*

Displays a description of the FEP files on the local host. The :Host keyword allows you to specify another host.

keywords :Format, :Highlight Files In Use, :Highlighting Mode, :Host, :More Processing, :Output Destination, :Type, :Unit

:Format {Normal, Detailed} How much information to include in the display. The default is Normal, meaning file name, length in blocks, and file comment (if any) are displayed for each file. Detailed means that all information, including creation date and author, is displayed.

:Highlight Files In Use
{Yes, No} Whether to indicate files that are currently in use. The default is Yes. This keyword works only when displaying the FEP file system of the local host.

:Highlighting Mode
{Bold, Arrow} How to indicate that a file is in use. Bold means

display the filename in boldface, Arrow means prefix the filename with an arrow. (The arrow is useful from a remote terminal.) The default is Bold except on remote terminals, where the default is Arrow.

:Host — A host on the network. The default is local.

:More Processing — {Default, Yes, No} Controls whether **More** processing at end of page is enabled during the output to interactive streams. The default is Default.

If No, output from this command is not subject to **More** processing.

If Default, output from this command is subject to the prevailing setting of **More** processing for the window. (See the section "FUNCTION M".)

If Yes, output from this command is subject to **More** processing unless it was disabled globally (see the section "FUNCTION M").

:Output Destination — {Buffer, File, Kill Ring, Printer, Stream, Window} Where to redirect the typeout done by this command. The default is the stream ***standard-output***.

:Type — {All, Boot, Fep, LMFS, Microcode, Other, Paging, World} The type of file(s) to display information about. The default is All. More than one type can be given, separated by commas.

:Unit — {*disk-unit-number* All} The default is All. *disk-unit-number* is an integer, interpreted as a disk unit number on the specified host.

Show FEP Directory first displays the number of free blocks and the proportion of blocks used on each disk unit. It then displays a summary of the files on each unit grouped by file type.

Show Herald **Command**

Show Herald *keywords*

Displays the herald message. The herald is a a multiline message displayed when you cold or warm boot, use the Command Processor (CP) Show Herald or Save World commands, or use the Disk Restore FEP command.

The herald shows you the name of the FEP file or partition for the current world load, any comment added to the herald, a measure of the physical memory and swapping space available, the versions of the systems that are running, the site's name, and the machine's own host name.

When you cold or warm boot, your machine displays a full-screen herald. When you display the herald with the Show Herald command, you see an abbreviated herald.

keywords :Detailed, :More Processing, :Output Destination

:Detailed {Yes, No, Normal} Whether or not to print the version information in full detail. Normal displays the systems which are usually of interest to users. Yes displays more systems than does Normal. No shows only machine information and release level, and no information about system versions. The default is Normal. The mentioned default is Yes.

:More Processing {Default, Yes, No} Controls whether **More** processing at end of page is enabled during the output to interactive streams. The default is Default.

If No, output from this command is not subject to **More** processing.

If Default, output from this command is subject to the prevailing setting of **More** processing for the window. (See the section "FUNCTION M".)

If Yes, output from this command is subject to **More** processing unless it was disabled globally (see the section "FUNCTION M").

:Output Destination {Buffer, File, Kill Ring, Printer, Stream, Window} Where to redirect the typeout done by this command. The default is the stream ***standard-output***.

For information about herald-related functions:

- See the function **sct:get-system-version**.
- See the function **sct:print-system-status-warning**.

si:enable-who-calls &optional *mode* *Function*

mode describes how the who-calls database should record the callers of any function. For more information about the **who-calls** database, see the section "Enabling the Who-Calls Database".

:all If you want to include callers of the Symbolics-supplied software (that is, software contained in the distribution world) in the database, use **:all**. This enables you to create the database once and then save it when you save the world. (When used with this argument, **si:full-gc** would discard the existing database and then remake it).

:all-remake Includes callers of the Symbolics-supplied and site-specific software in the database. Use this if you do not want to perform a **si:full-gc**. (When used with this argument, **si:full-gc** would discard the existing database and then remake it).

:new Enables the **who-calls** database to record the callers in any layered products, special software, or programs loaded into the world (after the site has been set). The Set Site command uses this argument by default. **:new** does not cause the callers of software in the distribution world to be recorded.

:all-no-make Enables the **who-calls** database to record the callers in any layered products, special software, or programs loaded into the world (after the site has been set), and does not cause the callers of software in the distribution world to be recorded until **si:full-gc** is performed. Once **si:full-gc** is performed, those callers (for software in the distribution world) are recorded.

:explicit If you want only explicitly-named files to be in the database, use the function **si:enable-who-calls** with the argument **:explicit**.

Note: Creating a full database takes a long time and about 2000 pages of storage.

si:full-gc &key *system-release* *Function*

Garbage-collects the entire Genera virtual memory environment, including some static areas. However, because static areas change slowly and are not likely to contain much garbage, use **gc-immediately** or the command Start GC :Immediately instead. See the section "Start GC Command". **si:full-gc** leaves the garbage-collector facilities in the state that it originally finds them, that is, with the same dynamic and ephemeral option settings.

If you use **si:full-gc**, call it with no arguments. The option **:system-release** is reserved for use by Symbolics. **si:full-gc** does an immediate, complete, nonincremental garbage collection as a preparation for immediately saving a world.

si:full-gc performs these operations:

- Resets temporary areas.
- Sets up the static areas to be cleaned up.
- Flips.
- Scavenges and flushes oldspace.
- Makes static areas static again.

It is not useful to perform an Incremental Disk Save (IDS) after running **si:full-gc**. Perform a complete disk save, instead.

Note: The Command Processor command Optimize World is the preferred high-level interface to the functions **si:full-gc**, **si:reorder-memory**, and **si:optimize-compiled-functions**. See the section "Optimize World Command".

Using the Initialization Lists invoked by si:full-gc

Two initialization lists, accessed through the **full-gc** and **after-full-gc** keywords to **add-initialization**, are run by **si:full-gc**. See the section "Introduction to Initializations".

si:full-gc runs the forms on the **full-gc** initialization list and then garbage-collects without multiprocessing (inside a **without-interrupts** form). The machine essentially "freezes" and does nothing but garbage collection for the duration. This operation takes 20 minutes or more, depending on the size of the world. After the garbage collection is completed, and before it reenables scheduling and returns, **si:full-gc** runs the forms on the **after-full-gc** initialization list.

full-gc is a system initialization list. You can add forms to it by using the **:full-gc** keyword in the list of keywords that is the third argument of **add-initialization**. The **full-gc** initialization list is run just before a full garbage collection is performed by **si:full-gc**. All forms are executed without multiprocessing, so the evaluation of these forms must not require any use of multiprocessing: they should not go to sleep or do input/output operations that might wait for something.

Typical forms on this initialization list reset the temporary area of subsystems and make sure that what is logically garbage has no more pointers to it.

sct:get-system-version &optional *(system* **"System"***)* *Function*

Returns three values. The first two are the major and minor version numbers of the version of *system* currently loaded into the machine. The third is the status of the system, as a keyword symbol: **:experimental**, **:released**, **:obsolete**, or **:broken**. *system* defaults to **System**. This returns **nil** if that system is not present at all.

For CLOE, it uses *name*, which may be a symbol, string or system denoting a system, and returns information about the corresponding system. The three returned values are the system major version number, the minor version number, and the system status (such as **:released** or **:experimental**). Note that this function is only available on the 386 side.

```
(get-system-version "FROB") =>
3
2
:experimental
```

si:optimize-compiled-functions &optional (*verbose* **t**) *Function*

This function can be run at any time. It takes one to two minutes to run. When functions are redefined (but not when they are first defined) old calls to the new function will not be as fast as they originally were. This function causes the references to be "snapped out," that is, forward pointers are replaced by what they point to. This makes the references as fast as they were originally.

Because redefined functions are in a different place in memory than the original function, paging performance can be degraded when many patches have been loaded. This problem is not fixed by **si:optimize-compiled-functions**, but it is by **si:reorder-memory**.

si:optimize-compiled-functions is called automatically when needed by disk saving, or when calling **si:full-gc** or **si:reorder-memory**, so it will not usually be needed on its own. If, however, a new definition of a system is loaded over an old one, with no intention of saving the resulting world, running **si:optimize-compiled-functions** will yield a substantial performance improvement.

Note: The Command Processor command Optimize World is the preferred high-level interface to the functions **si:full-gc**, **si:reorder-memory**, and **si:optimize-compiled-functions**. See the section "Optimize World Command".

sct:print-system-status-warning &optional (*system* "**system**") *Function*

If *system*'s status is **:experimental**, prints out a warning reminding the user to load patches. If *system*'s status is **:broken**, prints out a warning cautioning the user that the system may not work. Otherwise, it does nothing.

si:reorder-memory &key (*incremental* **t**) (*run-without-interrupts* **t**) *Function*

This function can be run after **si:full-gc**. It moves objects around in memory in a manner that optimizes paging performance. While it is running, the machine cannot be used for any other purpose. Usually this function is invoked with the intention of immediately saving the world.

You should use **si:reorder-memory** on IDS worlds only if the parent world has itself been optimized or if the parent world does not need optimizing. If you violate this rule, your IDS worlds will be considerably larger than they need to be. If you follow the rule, **si:reorder-memory** will not increase the size of the IDS world.

Distribution worlds are already optimized. Site-configured worlds which have only had Set Site done on them (and no systems loaded) do not need to be optimized.

Note: The Command Processor command Optimize World is the preferred high-level interface to the functions **si:full-gc**, **si:reorder-memory**, and **si:optimize-compiled-functions**. See the section "Optimize World Command".

Logical Pathnames

A logical pathname is one that does not correspond to any particular physical file system on a host. Logical pathnames make it easy to keep software on more than one type of file system.

For example, the set of files containing the Symbolics system sources and online documentation system is stored at each site. Some sites store these files on a Lisp Machine File System (LMFS), others store them on a VAX/Berkeley UNIX host with the Chaosnet package, and still others use a UNIX host running NFS in a `.sct` directory (see the section "Using SCT with a UNIX File System"). It is also possible to use a VAX/VMS file server running DNA. More information is available about using VAX hosts. See the section "Site Configuration and Namespace Service".

Symbolics software uses logical pathnames. All sites create a logical host (called *SYS*). Logical pathnames and the logical SYS host allow software to work correctly (and the same way) at every site. All pathnames for system software files are logical, and all begin with the logical host SYS. Only the translation of each logical pathname to a physical pathname differs at each site.

The translation of logical to physical pathnames depends on the translations files loaded into the current world. For more information about the translations files, see the section "Pathname Translation".

A site that stores the system software on a UNIX system translates logical pathnames into UNIX pathnames. A site that stores the system software on a LMFS translates logical pathnames into LMFS pathnames.

The flexibility of logical pathnames enables sites to split their logical SYS host across several physical hosts. A given physical host might contain some of the system software, but the logical entity called a SYS host contains all of it.

Syntax for Logical Pathnames

A logical pathname has the form

```
HOST: DIRECTORY; NAME.TYPE.VERSION
```

In logical pathnames, dots separate the filename, type, and version. There is no way to specify a device within a logical pathname. When a logical pathname is parsed, a pathname is returned whose device component is **:unspecific**. Logical pathnames can be hierarchical; use semicolons to separate directory levels.

Logical pathnames can also be relative. That is, they can contain a directory component whose meaning is "when merging against a default, append this". The syntax for this is

```
HOST: ; DIRECTORY; NAME.TYPE.VERSION
```

Notice the semicolon [;] that is placed before the directory component. The previous pathname, merged against a default of

```
HOST: USER; FOO.LISP.NEWEST
```

would yield this:

```
HOST: USER; DIRECTORY; NAME.TYPE.VERSION
```

The equivalence-sign character (≡) can be used for quoting special characters such as spaces and semicolons. (The use of this character is discouraged, however, as

files named using it will probably not be transportable). The double-arrow character (↔) can be used as a place-holder for unspecified components. The **:newest**, **:oldest**, and **:wild** values for versions are specified with the strings NEWEST, OLDEST, and * respectively. On input, **:newest** can be represented by > and **:oldest** by <.

There is no init file naming convention for logical hosts; you cannot log in to them. The **:string-for-host**, **:string-for-wholine**, **:string-for-dired**, and **:string-for-editor** messages are all passed on to the translated pathname, but the **:string-for-printing** is handled by the **fs:logical-pathname** flavor itself and shows the logical name.

Wildcard Matching in Logical Pathnames

The system can match any directory or subdirectory, at any level. For example, you can ask the Show Directory command to list all font files anywhere in the SYS hierarchy like this:

```
Show Directory SYS:FONTS;**;*.BFD.*
```

Wildcards in logical pathnames correspond to the >**> syntax for LMFS pathnames, the [name...] syntax for VAX/VMS file specifications, and the /**/ syntax in UNIX file specifications. See the section "LMFS Pathnames". This makes it easy to specify logical pathname translations on Symbolics computers, VAX/VMS, and UNIX. For example:

```
;;; -*- Mode: LISP; Syntax:  Common-lisp; Package:  USER -*-

(fs:set-logical-pathname-host "SYS" :translations
   '(("**;" "ACME-SMBX:>Rel-8-0>sys>**>")))

(fs:set-logical-pathname-host "SYS"
  :translations
  '(("SYS:**;*.*.*" "ACME-VMS:SYMBOLICS:[REL8-0...]*.*;*"))
  :no-translate nil)
```

Consider this example:

```
;;; -*- Mode: LISP; Syntax:  Common-lisp; Package:  USER -*-

(fs:set-logical-pathname-host "SYS" :translations
  '(("**;" "ACME-VMS:[SYMBOLICS.REL-8-0.SYS...]")))
```

Consider the following UNIX example:

```
;;; -*- Mode:  LISP; Syntax:  Common-lisp; Package: USER -*-

(fs:set-logical-pathname-host "SYS" :translations
  '(("**;" "ACME-UNIX:/usr/share/symbolics/rel-8-0/sys.sct/**/")))
```

Note: Wherever a double asterisk [**] appears in a logical-host's pathname, a corresponding "wild-inferiors" pathname must exist in the physical-host's pathname.

For more information about LMFS and VAX/VMS pathnames, see the section "LMFS Pathnames" and see the section "VAX/VMS Pathnames".

Loading System Definitions

Once you have written a large program and defined it as a system, use the function **load-system** (or the Command Processor (CP) commands Compile System and Load System) to compile and load the system (plus any patches related to it).

For information about the function **load-system**, see the function **load-system**. For information about the Load System and Compile System Command Processor (CP) commands:

- See the section "Compile System Command".
- See the section "Load System Command".

Loading System Definitions Using Logical Pathnames

So that your system definition can use logical pathnames, create these files:

System File
: This file (named sys:site;*system-name*.system) contains a pointer to the system declaration file (defined within this section). The system file enables the **load-system** function to find and load your system (so that others can easily use it).

  If yours is an experimental or private system, you may not require a separate sys:site;*system-name*.system file. Instead, compile the **defsystem** in an editor buffer (or put a form that loads the system declaration in your initialization file).

  For more information about system files, see the section "System Files".

Translations File
: This file (named sys:site;*logical-host*.translations) describes each logical host defined in the current world. When you transport a world load to a new site, the translations file is reloaded from the site's sys:site; directory, and the site's logical pathnames are mapped into the appropriate, corresponding set of physical pathnames.

  For more information about translations files, see the section "Translations Files".

System Declaration File
: This file (named *logical-host:logical-directory;system-name*.lisp or *logical-host:logical-directory*;sysdcl.lisp) contains the **defsystem** for a system. For more information about system declaration files, see the section "System Declaration Files".

System Files

System files (named sys:site;*system-name*.system) enable the function **load-system** (which looks in the sys:site; logical directory) to identify a system name that is undefined in your environment. For example, if you type the following at a Lisp Listener:

```
Load System graphic-lisp
```

the **load-system** function looks for the file SYS:SITE;GRAPHIC-LISP.SYSTEM.

The system file must contain this form:

(**sct:set-system-source file** *"system-name"*
"logical-host:logical-directory;system-declaration-file")

If a logical host other than "sys" is needed, use the additional form:

(**fs:make-logical-pathname-host** *"logical-host"*)

For example, for the system **graphic-lisp**, the file SYS:SITE;GRAPHIC-LISP.SYSTEM contains the following:

```
;;; -*- Mode: LISP; Package: USER -*-

(fs:make-logical-pathname-host "graphic-lisp")
(sct:set-system-source-file "graphic-lisp"
                            "graphic-lisp: graphic-lisp; glisp-sys")
```

The first form, a call to **fs:make-logical-pathname-host**, defines a logical host. Commonly, the *"logical-host"* has the same name as *"system-name"*. **fs:make-logical-pathname-host** also loads the translations file, which defines the translation from logical pathnames to physical pathnames.

Make sure that **fs:make-logical-pathname-host** is the first form in the file, as the second form depends on having the logical host defined already. **sct:set-system-source-file** specifies the logical pathname of the system declaration file. **load-system**, after referring to the translation definitions, loads the system declaration file.

Translations Files

Translations files (named sys:site;*logical-host*.translations) define the translations from logical directories (on the logical host) to physical directories (on a physical host). A translations file looks like this:

```
(fs:set-logical-pathname-host "logical-host"
 :physical-host "host-name"
 :translations '(("logical-directory;" "physical-directory"))
```

For example, for the system **graphic-lisp**, the file graphic-lisp.translations contains the following:

```
;;; -*- Mode: LISP; Package: USER -*-

(fs:set-logical-pathname-host "graphic-lisp"
  :physical-host "puzzle"
  :translations '(("graphic-lisp;" ">sys>graphic-lisp>")))
```

Notice the translations list in the previous example; the list consists of two-element lists (strings) that represent the logical directories specified in the system declaration and their associated physical directories.

To specify a hierarchy of directories (instead of a one-to-one translation), change the translations list as follows (where the double asterisk [**] means include all subdirectories of "graphic-lisp;"):

```
  :translations '(("graphic-lisp;**;" ">sys>graphic-lisp>**")))
```

In simple applications, where all system files are stored in one directory, it is common for the logical directory name (for example, "graphic-lisp;") to be the same as the system name ("graphic-lisp").

The sys:site;*logical-host.translations* file is loaded by **fs:make-logical-pathname-host**. Use **load-patches** to reload the file in the event that it has been changed.

System Declaration Files

System declaration files contain a **defsystem** form for defining your system and, if you need one, a **zl:defpackage** form (which must precede the system declaration). Any user-defined **defsystem** transformations should also precede the system declaration within this file.

Currently, a system declaration file can contain no more than one **defsystem** form, although any number of **defsubsystem** forms can appear in the file. This constraint exists because the system declaration can potentially be reloaded for each **defsystem** present (a situation difficult for the System Construction Tool (SCT) to resolve).

More information is available about **defsystem**, **zl:defpackage**, and **defsubsystem**.

- See the function **defsystem**.
- See the special form **defpackage**.
- See the function **defsubsystem**.

Here is a sample system declaration file:

```
;;; -*- Mode: LISP; Package: CL-USER; -*-
;;; Fortran package specifications
(defpackage fortran-global
  (:use)
  (:nicknames fortran for)
  (:prefix-name "FORTRAN")
  (:colon-mode :external)
  (:size 200))

(defpackage fortran-system
  (:use)
  (:nicknames for-sys)
  (:prefix-name "FOR-SYS")
  (:colon-mode :external)
  (:size 200))

(defpackage fortran-compiler
  (:use fortran-system fortran-global symbolics-common-lisp)
  (:nicknames for-compiler)
  (:prefix-name "FOR-COMPILER")
  (:colon-mode :external)
  (:size 1500))

(defpackage fortran-user
  (:use fortran-global symbolics-common-lisp)
  (:nicknames for-user)
  (:prefix-name "FOR-USER")
  (:relative-names-for-me (fortran-global user))
  (:size 2000))

;;; System definition using SCT
(defsystem fortran
    (:default-pathname "sys: fortran;"
     :journal-directory "sys: fortran;"
     :patchable t)
  (:module macros ("macros") (:root-module nil))
  (:module language-tools (language-tools) (:type :system))
  (:module front-end (fortran-front-end) (:type :system))
  (:module back-end (fortran-back-end) (:type :system))
  (:serial macros language-tools front-end back-end))

;;; Component system definition
(defsubsystem language-tools
    (:default-pathname "sys: language-tools;")
  (:serial ... ))
```

```
;;; Subsystem definition (non-patchable)
(defsubsystem fortran-front-end
    (:default-pathname "sys: fortran;")
  (:serial "tokenizer" "grammar" ... ))

;;; Subsystem definition (non-patchable)
(defsubsystem fortran-back-end
    (:default-pathname "sys: fortran;")
  (:serial "code-generator" "optimizer" ... ))
```

Since you specify the pathname explicitly with the form **sct:set-system-source-file** (inside the system file), system declaration filenames do not require an exact format. Typically, though, the logical pathname for them is *logical-host:logical-directory;system-name*.

Give the system declaration source file the lisp canonical file type. When you call the **load-system** function, **sct:set-system-source-file** loads the system declaration file (.newest version).

Loading System Definitions Using Physical Pathnames

To load system definitions that use physical pathnames, specify the name of the system and the pathname of the system declaration file in an **sct:set-system-source-file** form. Have your init file evaluate the form (or type the form at a Lisp Listener) prior to calling the function **load-system**. For more information about the function **sct:set-system-source-file**, see the section "Lisp Functions for Loading System Definitions".

Note: Logical pathnames enable you to change only translations (instead of editing all of your files to contain new file names) when moving programs between hosts (that use different operating systems, for example). Use logical pathnames — rather than physical pathnames — to ensure the site-independence of your systems.

Lisp Functions for Loading System Definitions

The Lisp functions described within this section are especially useful for site maintainers who make and distribute worlds.

sct:set-system-source-file *system-name source-file* *Function*

Specifies the pathname (*source-file*) of a file containing the system declaration for a system called *system-name*. Although **sct:set-system-source-file** can be used in two ways, Symbolics recommends the first.

1. When your system is defined with logical pathnames, include the **sct:set-system-source-file** form in the file sys:site;*system-name*.system. **load-system**

loads the sys:site;*system-name*.system file the first time you attempt to load the system.

2. When your system is defined using physical pathnames, have your init file evaluate the **sct:set-system-source-file** form (or type the form at a Lisp Listener) prior to calling **load-system** or to using the Load System or Compile System Command Processor (CP) commands. *Source-file* is loaded the first time you compile or load your system.

More information is available about using the function **sct:set-system-source-file** in system files. See the section "System Files".

fs:set-logical-pathname-host *logical-host* &key *:physical-host :translations :rules :site-rules (:no-translate* **t***) :no-search-for-shadowed-physical* *Function*

Creates a logical host named "*logical-host*" if one does not already exist. This form appears in `sys:site;`*logical-host*`.translations` files. It establishes the translations of logical directories on *logical-host* to physical directories on one or more physical hosts. The machine specified by the *:physical-host* keyword serves as the default physical host.

The *:translations* keyword specifies the list of translations from logical to physical directories.

- For more information about translations lists: See the section "Translations Files".

- For the format of the lists and the translation rules: See the section "Pathname Translation".

- For a discussion of the *:rules* and *:site-rules* keywords: See the section "Defining a Translation Rule".

If *no-translate* is **nil**, the translation of every interned logical pathname is checked. Properties are copied from the old physical pathname to the the new one, and logical pathnames that now have no corresponding physical pathnames are uninterned.

If *no-translate* is not **nil** or not supplied, this mapping is suppressed, and some physical pathnames might not get the properties of the logical pathname. This is not normally of any consequence, so *no-translate* defaults to **t**.

The argument *no-search-for-shadowed-physical* (default **nil**) means to look only in the existing pathname hosts for a host with the same name as the logical host. This saves time by not asking the namespace server whether the name of the newly defined logical host conflicts with the names of any physical hosts, but it prevents you from seeing the following warnings:

```
Warning: the host ~A must now be referred to as ~A: in pathnames,
                        since ~A is now a logical pathname host.
                        This affects ~[no~:;~:*~D~] extant pathnames.

Warning: the nickname ~A: for the physical host ~A
                          will now refer instead to the
                          logical pathname host ~A.
                          Use ~A: in pathnames.
```

For more information about sys.translations files, see the section "Pathname Translation". Also see the section "Translations Files".

fs:make-logical-pathname-host *name* &key *no-search-for-shadowed-physical*
Function

Defines *name* (a string or symbol) to be the name of a logical pathname host. *Name* should not conflict with the name of any existing host, logical or physical. An **fs:make-logical-pathname-host** form often appears in the file sys:site;*system-name*.system.

fs:make-logical-pathname-host loads the file sys:site;*name*.translations. **load-patches** checks the translations file for each logical host that is defined in the current world; if any translations file has been changed it is reloaded (if and only if no specific systems are specified in its arguments).

The argument **:no-search-for-shadowed-physical** (default **nil**) means to look only in the existing pathname hosts for a host with the same name as the logical host. This saves time by not asking the namespace server whether the name of the newly defined logical host conflicts with the names of any physical hosts, but it prevents you from seeing the following warnings:

```
Warning: the host ~A must now be referred to as ~A: in pathnames,
                        since ~A is now a logical pathname host.
                        This affects ~[no~:;~:*~D~] extant pathnames.

Warning: the nickname ~A: for the physical host ~A
                          will now refer instead to the
                          logical pathname host ~A.
                          Use ~A: in pathnames.
```

Note: fs:add-logical-pathname-host is an obsolete name for this function.

More information is available about using the function **fs:make-logical-pathname-host** in system files. See the section "System Files".

Pathname Translation

A translations file contains the form **fs:set-logical-pathname-host** and a translations list. The list describes logical pathnames by providing their corresponding

physical pathnames. Each logical/physical pathname pair is called a translation pair.

The Logical Pathname Translations File

Here is a sample translations file (note the translations list, containing three translation pairs, following the `:translations` keyword at the end of the file):

```
;;; -*- Mode: LISP; Package: FS; Syntax: ZetaLisp; Base: 10; -*-
(fs:set-logical-pathname-host "SYS"
   :physical-host "ACME-YUKON"
   :translations '(("SYS:DOC;**;*.*.*" "ACME-YUKON:>sys>doc>**>*.*.*")
                   ("SYS:FONTS;**;*.*.*" "ACME-RIVERSIDE:>sys>fonts>**>*.*.*")
                   ("SYS:**;*.*.*" "ACME-QUABBIN:>sys>**>*.*.*")))
```

In this sample, the logical pathname `SYS:DOC;` (and all its inferior directories) maps to the physical host ACME-YUKON. If this translations file were loaded, the logical pathname,

```
SYS:DOC;SITE;SITE7.SAB.NEWEST
```

would resolve to the file described by this physical pathname:

```
ACME-YUKON:>sys>doc>site>site7.sab.newest
```

Likewise, the logical pathname, `SYS:FONTS;` (and all its inferior directories) would map to the physical host ACME-RIVERSIDE, and the logical pathname,

```
SYS:FONTS;TV;CPTFONTB.BFD.NEWEST
```

would resolve to the file described by this physical pathname:

```
ACME-RIVERSIDE:>sys>fonts>tv>cptfontb.bfd.newest
```

All other logical pathnames beginning with `SYS:` (and all their inferior directories) would map to the physical host ACME-QUABBIN. For example,

```
SYS:FLAVOR;CTYPES.LISP.NEWEST
```

would resolve to the file described by this physical pathname:

```
ACME-QUABBIN:>sys>flavor>ctypes.lisp.newest
```

The Logical Pathname Translation Process

There are two phases to the translation process. In the first phase, using the **:pathname-match** message, the pathname resolver matches a logical pathname (the pathname to be translated) against the logical pathnames listed successively in the translations file. Once the pathname resolver finds a match, it uses the appropriate logical/physical pathname pair from the list.

Note: Because the translation list is searched in sequence, it should provide the most specific pathnames first, and the most general pathname last.

In the second phase, the pathname resolver processes the selected translation pair according to translation rules. There are three sets of translation rules for each logical host:

Permanent	The permanent translation rules are special purpose rules that cannot be overridden. They provide for such things as the translation of patch file pathnames. This set is searched first.
Site	The site translation rule is determined by the Site-Directory attribute for each object of class "site" in the namespace. A site's translation rule cannot be overridden. This set is searched second.
Supplied	The normal, supplied translation rules are provided by the author of the software using the logical host. This set is searched third.

Additionally, the pathname resolver uses these host-independent rules:

Global	This set is not currently used for anything, but it is provided for future extension.
Default	This is the **:translate-wild** rule, which uses **:translate-wild-pathname-reversible**. This rule is used when no other rule is applicable.

The second phase (in which the the pathname resolver processes the selected translation pair according to translation rules) is potentially more complex. In its simplest form, the pathname resolver uses the default rule. Before using the default rule, though, the pathname resolver searches for a more suitable one.

The default rule produces a physical pathname by sending the **:translate-wild-pathname-reversible** message to the logical pathname, where the first element of the translation pair is the source pattern, and the second element of the translation pair is the target pattern. For more information about source and target patterns:

See the section "Wildcard Pathname Mapping". See the section "Wildcard Directory Mapping". See the section "Reversible Wildcard Pathname Translation".

Logical Translations to Multiple Physical Hosts

A logical host can translate to more than one physical host when the translations list given to **fs:set-logical-pathname-host** contains explicit pointers to more than one host.

For example:

```
(fs:set-logical-pathname-host "SYS"
  :translations '(("SYS:DOC;**;*.*.*" "ACME-LISPM:>Rel-8-0>doc>**>*.*.*")
                  ("SYS:**;*.*.*" "ACMEVAX:SYMBOLICS:[REL8-0...]*.*.*"))
  :no-translate nil)
```

Note: it is not necessary to specify the **:physical-host** argument to **fs:set-logical-pathname-host** as long as host names are specified in the translations list. If a **:physical-host** argument is specified, however, it serves as the default.

The Front-End Processor

When you boot a Symbolics computer, you communicate with the Front-End Processor (the FEP). The FEP loads those files that enable the local machine to boot. On Symbolics 3600-family machines, the FEP is a separate chip and takes care of other needs, such as listening for the machine's keyboard and mouse.

These components make up the FEP:

1. A microprocessor.

2. A FEP Kernel.

3. Overlay files (also called "flods") containing loadable software. The FEP can read these from tape or disk. Overlay (flod) files provide support for new features, and are supplied as part of each new Genera release.

On Ivory-based machines, there is no separate processor; the FEP is implemented in software. The FEP kernel resides on disk (FEP kernel version I307 or greater).

On 3600-family machines, the FEP kernel resides in EPROM (EPROM version 127 if you have a 3640 or 3670 machine; EPROM version 206 if you have an 3610, 3620, or 3650 machine manufactured before the release of Genera 7.2; EPROM version 208 if you have a 3620, 3630, or 3650 machine manufactured after the release of Genera 7.2).

Use the Show Version FEP command to determine the FEP (EPROM or software) version with which your machine has been equipped. (For information about the Show Version FEP command, see the section "Show Version FEP Command".)

If you have a Symbolics 3600-family machine, and it is equipped with an EPROM whose version number is lower than 127, please contact Symbolics Customer Service for an upgrade.

Overlay (Flod) Files and the FEP

The FEP implements some basic commands from its kernel (the FEP kernel is resident in EPROM or software, depending on what type of machine you have). Kernel commands include startup commands and the display and disk drivers, for example. Additional, release-specific commands reside in loadable software, specifically in the FEP overlay (flod) files, loaded onto your machine with the Copy Flod File command. All flod files have the extension ".flod", which identifies them as overlays. See the section "Copy Flod Files Command".

In order for the FEP to use them, each overlay file must be paged into memory. Only one overlay file at a time can be memory-resident, and each overlay file must be scanned before the FEP can access — and use — the commands within it.

Scanning inserts those commands defined in an overlay file into the FEP's command tables. Once scanned, these commands remain in the FEP's command tables until you reset the FEP, or power down your machine.

When you type a FEP command, three things can happen:

1. The command is resident or the correct overlay has been scanned and paged in; the FEP immediately executes the command.

2. The command resides in an overlay that has been scanned but not yet been paged in; the FEP pages in this overlay, scans it, and executes the command.

3. The command resides in an overlay that has not been scanned.

Here is a list of the overlay (flod) files and some examples of the types of commands contained within them. To read this list, replace the wildcard symbol (*) with the FEP EPROM version (for example, V127, G206, or G208) or Ivory FEP kernel version (for example, I315) for your machine:

Overlay File:	*Contains:*
*-info.flod	Commands that give information about the machine, such as the Show Configuration command.
*-loaders.flod	Commands to load the machine, such as the Load Microcode and Load World commands.
*-lisp.flod	Commands for manipulating Lisp, such as the Start, Continue, and Show Status commands.
*-debug.flod	The FEP Debugger, which is invoked by the Debug command.
*-tests.flod	The Test commands.
*-disk.flod	The Disk Restore and Disk Format commands.

The last two overlay files are used only during software installation or testing. More information is available about using the *-tests.flod and *-disk.flod files. See the section "Scanning the Overlay (Flod) Files".

Saving Previous FEP Kernels and FLOD Files

Some users like to "houseclean" their FEP-related files, and delete all but the most recent version. This is a dangerous habit, and we recommend against it. Backup versions of FEP files are necessary in some debugging situations. Since FEP files (both the kernel and flod files) do not require much disk space, we recommend that you save the previous versions of these files.

Ivory users in particular should save the previous version of the FEP kernel and flod files. Note that on Ivory machines, there is no FEP in PROM as there is on 3600-family machines, so your options are very limited if you depend on a single copy on disk and it goes bad.

The system attempts to prevent you from deleting the previous kernel, but it does not keep you from deleting the previous flods. The kernel is not much use without its flod files.

Scanning the Overlay (Flod) Files

Use the hello.boot file to scan the overlay (flod) files so that the FEP can use the commands contained within them. The FEP command Hello loads the hello.boot file for you.

If one does not already exist, create a hello.boot file in the editor, with a pathname of FEPn:>hello.boot (FEP*n* refers to the disk unit number where the hello.boot file resides, if your computer has more than one disk associated with it).

This file should contain a sequence of commands to scan the overlay (flod) files (except for those overlays containing special commands used for installation and testing). Hello.boot should also contain the command Initialize Hardware Tables. Additionally, the hello.boot file should contain other commands that you want your machine to execute every time you boot. (This is in contrast to the boot.boot file, which should contain commands specific to the world you load.)

Here is a sample hello.boot file for a 3600-family machine:

```
Scan v127-info.flod
Scan V127-loaders.flod
Scan v127-lisp.flod
Scan v127-debug.flod
Initialize Hardware Tables
Declare Paging-Files FEP0:>Paging-1.page
Declare More Paging-Files FEP0:>Paging-2.page,Paging-3.page
Set Chaos-Address 52525
```

Here is a sample hello.boot file for an Ivory-based machine:

```
Hello Innn
Hello Local (or hostname)
```

I*nnn* and Local represent two .boot files. Their contents should be as follows:

Hello I*nnn* Boot File

The I*nnn*.boot file (where *nnn* is the IFep version number, which is 325 for Genera 8.1) should contain the commands to scan the flod files and initialize things.

```
Scan I325-lisp.flod
Scan I325-loaders.flod
Scan I325-info.flod
Scan I325-debug.flod
Initialize Hardware Tables
```

Hello Local Boot File

The local.boot file should contain those commands that set up this specific machine, declaring paging files, setting the network address, and any other boot options.

```
Declare Paging Files FEP0:>Paging-1.page
Declare More Paging Files FEP0:>Paging-2.page,Paging-3.page
Set Boot Options :Network Address Chaos|52525 :IDS Enable
```

Use the Show Version FEP command to determine the FEP (EPROM or software) version with which your machine has been equipped. (For information about the Show Version FEP command, see the section "Show Version FEP Command".)

If you have a Symbolics 3600-family machine, and it is equipped with an EPROM whose version number is lower than 127, please contact Symbolics Customer Service for an upgrade.

Make sure you press RETURN after the last command, and then save the file. For an explanation of the Scan commands, see the section "Overlay (Flod) Files and the FEP".

When necessary (before issuing installation and test commands), use the Scan command to explicitly page in and scan the -tests.flod and -disk.flod files. Instead of using a hello.boot file, type the following to the FEP prompt (the asterisk (*) represents the EPROM version present in your machine):

```
Scan *-disk.flod
Scan *-tests.flod
```

For more information about the FEP file system and FEP files (such as the hello.boot file) see the section "FEP File Systems".

Using Lisp to Write Overlay (Flod) Files to Cartridge Tape

The Lisp function **tape:write-fep-overlay-flods-to-cart** writes the overlay (flod) files to a cartridge tape in the appropriate format for the FEP's Scan command. We recommend that you use this function to make a backup tape containing overlay (flod) files. If you ever find yourself without flod files on disk, you can use the backup tape to get them.

This example shows how to copy v127 flod files to tape:

```
(tape:write-fep-overlay-flods-to-cart "V127")
```

To use the backup tape, load it into a tape drive and type the following at the FEP prompt:

```
FEP Command: Mount Cart:
FEP Command: Scan Cart:
FEP Command: Scan
.
.
.
```

Repeat the Scan command until you get an "End of File" notification. Then, type boot to activate the boot file, or manually type each boot command at the prompts. For more information about manual booting, see the section "Booting the Symbolics Machine".

Once you've booted Lisp, copy the overlay files from SYS:N-FEP; onto the FEP file system. Use the Copy Flod Files command to do this.

- For information about what each overlay file contains, see the section "Overlay (Flod) Files and the FEP".
- For information about using the Copy Flod Files command, see the section "Copy Flod Files Command".
- For information about using this Lisp function, see the function **tape:write-fep-overlay-flods-to-cart**.
- For information about using the debug.flod files, see the section "Debugging in the FEP".

FEP File Systems

Although FEP is an acronym for *Front-End Processor*, FEP file systems are managed primarily by Genera. The FEP can only access files stored within the FEP file system on the local host. For example, the FEP needs to use FEP file systems to boot the machine, and run diagnostics. FEP file systems are also used to organize and store files that are needed for system overhead (such as paging files).

FEP file systems support multiple file versions, soft deletion, and expunging. They also use hierarchical directories.

The need to allow the FEP to access FEP files — while at the same time, allowing the rest of the system to use them — imposes these constraints on the design of FEP file systems:

- The internal data structure of files within FEP file systems must be simple enough to permit the FEP to read them.
- A small amount of concurrent access by both the FEP and Lisp must be allowed.
- A FEP file's data blocks need a high degree of locality on the disk, to minimize access time.
- FEP file systems must be reliable; the FEP needs to use them for basic operations, such as the running of diagnostics, and the booting of each machine.

Allocating new blocks for FEP files is a slow process. The creation of many files (especially small ones) can impair system performance because essential files — such as those used for paging or world loads — can become inappropriately fragmented.

To see the contents of a FEP file system, use the Command Processor (CP) Show FEP Directory or Show Directory commands. Alternatively, use m-x Dired, the Command Processor (CP) Edit Directory command, or the File System Editor Activity (FSEdit).

For more information about how to use the Show FEP Directory command, see the section "Show FEP Directory Command".

For more information about how to use Dired, see the section "Dired/Edit Directory".

For more information about how to use FSEdit, see the section "Using FSEdit".

FEP File Systems and Symbolics Computer Disks

Symbolics computers can have more than one local disk, and each machine's FEP can access all of them. Currently, hardware limits the maximum number of any one 3600-family machine's disks to eight. MacIvory machines have a maximum peripheral device limit of seven.

The form FEP: refers to the disk (by default) from which the current world was booted. Disk 0 is usually the default, so typing FEP: is usually equivalent to typing FEP0: . Besides using the default, you can specify disks explicitly, using forms such as FEP1: or FEP7: .

Here are some activities that require you to specify a disk unit number (explicitly or implicitly):

- Booting a world with the Load World FEP command.
- Adding paging files with the Declare Paging Files FEP command at boot time,

  or

- Adding paging files with the Command Processor (CP) Add Paging File command from an already-running Lisp world.

Note: The `.ilod` file extension indicates world-load files for Ivory-based machines, just as the `.load` file extension indicates world-load files for Symbolics 3600-family machines. Files with the `.ilod` extension can be copied only between Ivory-based machines. Files with the `.load` extension can be copied only between Symbolics 3600-series machines.

FEP File Systems on 3600-Series and XL400 Systems

Each 3600-family or XL400 disk must have a FEP file system on it that describes the disk space available on it.

Each disk unit is presumed to contain one FEP file system. FEP file systems are named FEP*n* (where *n* is the disk unit number on which the FEP file system resides).

This scheme allows gaps in the sequence of FEP file system names. For example, a machine with disk unit 0 and disk unit 2 (but no disk unit 1) has FEP file systems named FEP0 and FEP2 (but none named FEP1).

FEP File Systems on MacIvory Systems

MacIvory systems share disk space (swap space) between the Ivory and Macintosh processors. In order for the Ivory processor to access a Macintosh's disk to find a world load, for example, that disk must have at least one Ivory partition on it. (Symbolics recommends that you limit the number of Ivory partitions on each disk to one.)

Ivory partitions represent disk space to which the Macintosh processor does not have access. Each Ivory partition must contain a FEP file system that describes the disk space available in it.

Each time you power up or boot a MacIvory, the system checks the disk from which you booted. Next, it checks the remaining disks, according to their respective Small Computer Serial Interface (SCSI) bus priorities.

The first Ivory partition that the system finds is presumed to contain the FEP file system named FEP0. Any remaining Ivory partitions are presumed to contain the FEP file systems named FEP1, FEP2, and so on. This scheme does not allow gaps in the sequence of MacIvory FEP file system names.

FEP Pathnames

FEP pathnames can include references to a host, disk-unit, directory, filename, file type, and version. Separate the host-name from the rest of a file pathname by using a vertical bar (you can see this in the example that follows).

Delimit FEP pathnames like this:

```
Picasso|FEP0:>directory-name>filename.type.version
```

Pathname components are:

Host — Specifies which machine's FEP file system you are referencing. The default is the local machine.

Disk-unit — Specifies the disk unit number on which the local host's FEP file system resides. The initial default is FEP0. Later, the default becomes the local disk unit from which the world was booted. Netbooting doesn't change the default. Symbolics suggests that you specify the disk-unit number explicitly, since the default may be different for different worlds.

Directory — Indicates the name of the FEP file system directory (directory names cannot exceed 32 characters). There is no limit on the total length of a hierarchical directory specification.

Filename Indicates the name of the FEP file (filenames cannot exceed 32 characters).

File type Indicates the type of the FEP file (file types cannot exceed 4 characters).

Version Indicates the version number of the FEP file (this must be a positive integer or the word "newest").

Note: Although you can access FEP files on other hosts from Genera, the FEP has access only to the local host.

For information about Ivory-based machines and the host pathname syntax for them, see the section "Accessing the Macintosh File System".

FEP File Types

By convention, the FEP file system uses the following extensions to deliniate file types:

boot Files with the .boot extension contain commands that can be read and executed by the FEP.

load Files with the .load extension contain a world load image, (sometimes called a band) for Symbolics 3600-family machines. Files with the `.load` extension can only be copied between Symbolics 3600-family machines.

ilod Files with the .ilod extension contain a world load image, (sometimes called a band) for Ivory machines. Files with the `.ilod` extension can only be copied between MacIvory, XL400, and Symbolics UX-family machines.

mic Files with the .mic extension contain a microcode image, plus the contents of other internal high-speed memories that are initialized when Symbolics 3600-family machines are booted. For example, >3640-mic.mic.428 contains version 428 of the microcode for 3640 and 3670 machines.

fspt In order to use the local Lisp Machine File System (LMFS), Lisp must have access to the File System Partition Table (FSPT). The File System Partition Table is contained within a FEP file named `fspt.fspt` that lists the LMFS partitions.

file Files with the .file extension are Lisp Machine File System (LMFS) partitions.

page Files with the .page extension are used exclusively as virtual memory swap space during the current boot session.

flod Files with the .flod extension are FEP overlay (flod) files. Such files contain binary code (FEP software).

fep

Files with the .fep extension are FEP-specific; they contain information about the organization of fep files on the disk.

Note: Since FEP-specific files are system files, not user files, they should not be written to by user programs.

>free-pages.fep
: This file describes which blocks on the disk are free.

>bad-blocks.fep
: This file lists all of the blocks that contain media defects.

>sequence-number.fep
: This file contains the highest sequence number in use. The FEP file system uses sequence numbers to uniquely identify files. (These help to rebuild the file system, should a catastrophic disk failure occur.)

>disk-label.fep
: This file contains the disk pack's physical disk label. The label is used to identify the pack, and to describe its characteristics.

>kernel.fep
: This file exists only on Ivory-based machines. It contains the FEP software which, on Symbolics 3600-family machines, resides in EPROM.

>reserve.fep
: This file is reserved for use by Symbolics software.

>unique-id.fep
: This file is reserved for use by Symbolics software.

dir

Files with the .dir extension are FEP subdirectories. Use the Show Directory FEP command or the Command Processor (CP) Show Directory Command to see the contents of FEP subdirectories. For more information about these commands,

- See the section "Show Directory FEP Command".
- See the section "Show Directory Command".

Note: Since the directory `>root-directory.dir` is a system file, not a user file, it should not be written to by user programs.

FEP File Properties

FEP file properties store information about FEP files (such as when they were last written, and whether they can be deleted).

File properties are read by the **fs:file-properties** function, and modified by the **fs:change-file-properties** function. The function **fs:directory-list** returns the file properties of several files at once.

The following file properties can be both read and modified:

:creation-date	The universal time at which a file was last written. (See the section "Dates and Times".)
:author	The user-ID of the last writer to a file: a string.
:length-in-bytes	The length of a file, expressed as an integer.
:deleted	When **t**, a file is marked as deleted. Disk space is not reclaimed until you expunge the directory in which a deleted file resides.
:dont-delete	When **t**, attempting to delete or overwrite a file signals an error. When **nil**, indicates that a file can be written to or deleted.
:comment	Written comments displayed in brackets: a string.

These file properties are returned by the **:properties** message. They cannot, however, be modified by **:change-properties**:

:byte-size	The number of bits in a byte. The value of this property is always eight.
:length-in-blocks	The block length of a file expressed as an integer.
:directory	When **t**, the file is a directory, otherwise **nil**.

Creating Free Space on the Local Disk

There are times when you'll need to create free space on the local disk (for example, when copying or save worlds, creating additional paging files, or to increase the size of a Lisp Machine File System (LMFS)). There are three ways to create free space on the local disk:

- Delete and expunge unneeded world load files.
- Delete and expunge unused paging files.
- Vacate and delete unnecessary LMFS partitions.

This section describes how to perform these procedures.

Deleting and Expunging Unneeded World Loads

Note: Unless netbooting service is available at your site, you must keep at least one base (distribution) world on your disk.

1. Check to see which world load files are in use. To do this, issue the Command Processor (CP) Show FEP Directory command and specify "world load files", like this:

   ```
   Show FEP Directory :type world
   ```

   This will display the world load files on the local disk; those in use will be shown in boldface type on your screen.

2. Delete and expunge any obsolete world load files. This will provide you with additional space. (If this doesn't provide you with enough disk space for your needs, go on to the next procedure, and delete and expunge unused paging files.)

Deleting and Expunging Unused Paging Files

1. Check to see whether all of your paging files are in use. To do this, issue the Command Processor (CP) Show FEP Directory command and specify "paging files" like this:

   ```
   Show FEP Directory :type paging
   ```

   This will display the paging files on the local disk; those in use will be shown in boldface type on your screen.

2. If all the paging files are currently in use, cold boot the machine, but do not declare any of the paging files that you intend to delete. Manually type these FEP commands on a 3600-family machine:

   ```
   Declare Paging-Files with no argument supplied
   Declare Paging-Files the-names-of-paging-files-you-want-to-keep
   Clear Machine
   Load Microcode
   Load World world-load-filename
   Enable IDS
   Set Chaos-Address this-machine's-Chaos-address
   Start
   ```

   Manually type these commands on an Ivory-based machine:

   ```
   Declare Paging Files with no argument supplied
   Declare Paging Files the-names-of-paging-files-you-want-to-keep
   Clear Machine
   Load World world-load-filename
   Enable IDS
   Set Network Address Chaos|this-machine's-Chaos-address
   Start
   ```

   Cold booting with this sequence of commands enables you to remove some paging files from use, and to declare a new list of paging files. For informa-

tion about how the Declare Paging Files FEP command works, see the section "Declare Paging Files FEP Command".

3. Delete and expunge the paging files that are not in use. This will provide you with additional space. Be sure to remove any references to the deleted paging files from your boot file(s), as well.

Vacating and Removing Unnecessary LMFS Partitions

Note: Auxilliary file partitions should be removed only by using the File System Editor (FSEdit).

1. Check to see if how many LMFS partitions are present. To do this, issue the Command Processor (CP) Show FEP Directory command and specify "LMFS files" like this:

```
Show FEP Directory :type LMFS
```

   This will display the LMFS-related files on the local disk. If only one file of type .file exists, this is your main LMFS partition; do not attempt to remove it. If more than one file of type .file exists, go to Level 2 of the File System Editor (FSEdit).

2. In Level 2 of the File System Editor (FSEdit), click right on the [Free Records] menu item to find out how much room is available in each LMFS partition. If one partition has enough space so that you can move the contents of another LMFS partition into it, go to Level 3 of the File System Editor (FSEdit).

3. In Level 3 of the File System Editor, click Left, Middle, or Right on the [Remove Partition] menu item. You will be queried about which partitions to vacate, remove, and delete.

4. Expunge the LMFS partitions that you have vacated, removed, and deleted.

Increasing Available Paging (Swap) Space

Programs that use large amounts of virtual memory might require the allocation of additional paging (swap) space. To create a 20,000 block paging file on disk unit 0, use the Command Processor (CP) Create FEP File command. At the `Command:` prompt in a Lisp listener, type:

```
Command: Create FEP File fep0:>page1.page 20000
```

The Command Processor (CP) Add Paging File command will enable you to use the paging file from within the Lisp environment. Alternatively, halt the machine and use the Add Paging File FEP command, and then use the Continue FEP command to return to Lisp.

If you modify boot files with the Declare Paging Files command, you won't have to manually add the paging files again (when you next boot the machine). Issue the

Declare Paging Files FEP command before the Load World FEP command inside a boot file.

More detailed information is available about the Declare Paging Files command. See the section "Declare Paging Files FEP Command".

FEP Commands

Different types of FEP commands exist. This section provides details about all of them. In order to make it easy for you to find and use information about FEP commands, we've grouped them by function. Unless otherwise noted, each FEP command is implemented on both 3600-family and Ivory-based machines.

Symbolics computer users need some FEP commands for the day-to-day operation of their Symbolics computers. For descriptions of these FEP commands, see the section "Commonly Used FEP Commands".

Some FEP commands pertain specifically to systems with color consoles. For descriptions of these FEP commands, see the section "Color Systems FEP Commands".

Other FEP commands are designed for System Administrators, who perform routine system maintenance and troubleshooting tasks. For descriptions of these FEP commands, see the section "Systems Maintenance FEP Commands".

The remaining FEP commands are "for expert use". Be careful when using these commands. *If you make a mistake, you can destroy the state of the loaded or saved Lisp system*. For descriptions of these FEP commands, see the section "FEP Commands for Systems Experts".

For information about FEP-related Command Processor (CP) commands, see the section "FEP-Related Command Processor (CP) Commands". For information about Lisp Functions that site maintainers will find useful, see the section "FEP-Related Lisp Functions".

Note: Some FEP commands are unique to Symbolics 3600-family machines, some are unique to Symbolics Ivory-based machines, and some are common to both. The syntax for commands used by both machines can be different, however. Many 3600-family machine FEP commands, for example, require hyphens (but Ivory FEP commands never have hyphens in them). Before issuing a FEP command, make sure that you know how to use it appropriately.

If your system doesn't recognize a FEP command, you may need to scan an overlay (flod) file to make the command available. Alternatively, your machine may not support that command. For information about the overlay (flod) files, see the section "Overlay (Flod) Files and the FEP".

Entering FEP Commands

Note: Ivory-based machines allow you to press `c-ABORT` to stop any long-running FEP operations (such as Load World). 3600-family machines do not. On 3600-family and Ivory-based machines, you can type any character to abort FEP typeout.

The FEP supplies you with default arguments and can provide you with documentation about its commands. The FEP command prompt, displayed when you are using the FEP, looks like this:

```
FEP Command:
```

You need type only enough of a FEP command to identify it uniquely, as shown here:

This Input	*Completes to*
`b RETURN`	`Boot`
`l w RETURN`	`Load World (default is FEP0:>Genera-8-0.ilod)`
`st RETURN`	`Start`

At the prompt, press the `HELP` key for a list (and descriptions) of all FEP commands.

Once you have typed a command name, press the space bar and then the `HELP` key for a list of all possible completions to that command.

You can insert parenthetical comments within or after FEP commands in hello files or boot files. Such comments are useful, for example, for providing identification about different boot files:

```
load world >World-1.load (contains geological survey programs)

load world >World-2.load (contains simulator)
```

Pathname Completion in the FEP

Two types of pathname completion exist in the FEP. When you press the `COMPLETE` or `HELP` key, the FEP attempts to complete the pathname you have supplied, augmenting your input as far as possible (until it runs into a potential conflict between two similar pathnames in the file system).

For example, at a 3600-family machine, if you type:

```
Load Microcode (default is FEP0:>3640-mic.mic) 3640 COMPLETE
```

the FEP system responds like this:

```
Load Microcode (default is FEP0:>3640-mic.mic) FEP0:>3640-
```

because the possibilities are:

```
FEP0:>3640-mic.mic
FEP0:>3640-fpa-mic.mic
```

Similarly, if you type:

```
Load World (default is ...) Inc HELP
```

the machine will display all the files that begin with Inc (such as those worlds that were created using the Incremental Disk Save command).

On Ivory-based machines, the FEP will not choose deleted (but unexpunged) files when performing pathname completion or choosing the approriate file when the version is .newest. You can force the Ivory FEP to use a deleted file (one that, for example, you have erroneously deleted) by specifying its version number explicitly.

By using this technique along with the Add World File FEP command, you can force the FEP to load a deleted world hierarchy (provided that it has not been overwritten or expunged). More information is available about this command. See the section "Add World File FEP Command".

Pathname Merging in the FEP

Like Lisp, the FEP merges pathnames against the default. You need specify only the fields (name, type, or version) that differ from the (context-dependent) default.

If either the filename or the type is given, the default version will always be `.newest`. Pathnames are not case-sensitive. Here are some examples of pathname merging in the FEP:

Default	*Input*	*Merged*
`FEP0:>3600-MIC.MIC`	`3600-FPA-MIC`	`FEP0:>3600-FPA-MIC.MIC`
`FEP0:>3600-MIC.MIC`	`fep1:`	`FEP1:>3600-MIC.MIC`
`FEP0:>3600-MIC.MIC`	`..428`	`FEP0:>3600-MIC.MIC.428`
`FEP0:>3600-MIC.MIC.428`	`3600-FPA-MIC`	`FEP0:>3600-FPA-MIC.MIC`
`FEP0:>3600-MIC.MIC`	`3600-mic..428`	`FEP0:>3600-MIC.MIC.428`

Commonly Used FEP Commands

These FEP commands can be typed in at the FEP command prompt, or executed from within a command file (such as a `.boot` file). For your convenience, we have arranged these commands in alphabetical order.

Add Paging File FEP Command

This command is implemented on both Symbolics 3600-family and Symbolics Ivory-based machines. Although Ivory-based machines do not use hyphens in their FEP command names, many 3600-family machine FEP commands do. Where there are differences, they are shown here.

Add Paging-File *filename* (for 3600-family machines)

Add Paging File *filename* (for Ivory-based machines)

Adds a file to be used for virtual memory swap space during the current boot session.

filename Filename of the file to be used for paging. *Filename* defaults to FEP*n*:>page.page.

The Add Paging File FEP command adds a paging file only for current use; this command does not add a paging file to the list of declared paging files (declared by the Declare Paging Files command or the Declare More Paging Files command). See the section "Declare Paging Files FEP Command". See the section "Declare More Paging Files FEP Command".

Load the Add Paging File FEP command by scanning the overlay (flod) file *-loaders.flod.

Boot FEP Command

Boot *filename*

Executes the commands specified in *filename*. (On the MacIvory, this is automatically done for the user whenever Lisp is started). See the section "Using the Genera Application on a MacIvory".

filename The name of a boot file; the default is the last filename given to either the Boot or Show File commands (initially, the default is >Boot.boot).

The Boot FEP command is resident in the FEP, so it needn't be loaded from an overlay (flod) file.

Clear Machine FEP Command

Clear Machine

Clears Lisp memory. It is optional on Ivory-based machines.

On 3600-family machines, use the Clear Machine FEP command before loading new microcode, and before using the Disk Restore or Disk Format FEP commands. Load the command by scanning the overlay (flod) file *-loaders.flod. This command commonly appears in boot.boot and autoboot.boot files.

Clear Paging Files FEP Command

This command is implemented on both Symbolics 3600-family and Symbolics Ivory-based machines. Although Ivory-based machines do not use hyphens in their FEP command names, many 3600-family machine FEP commands do. Where there are differences, they are shown here.

Clear Paging-Files (for 3600-family machines)

Clear Paging Files (for Ivory-based machines)

Clears the list of paging files automatically added by the Load World FEP command. It also clears the list of paging files added by the Add Paging File FEP command.

The Clear Paging Files FEP command does not clear the list of paging files declared by the Declare Paging Files or Declare More Paging Files FEP commands. To clear this list, issue the Declare Paging Files command without listing any files.

Load the Clear Paging Files FEP command by scanning the overlay (flod) file `*-loaders.flod`. This command commonly appears in files of the type `boot.boot`, `hello.boot`, and `autoboot.boot`.

Clear Screen FEP Command

Clear Screen

On 3600-family machines, clears the console's screen. On Ivory-based machines, it clears the Ivory FEP window.

The Clear Screen FEP command is resident in the FEP, so it needn't be loaded from an overlay (flod) file. This command commonly appears in files of the type `hello.boot` and `autoboot.boot`.

Continue FEP Command

Continue

Returns you to Lisp from the FEP.

On Ivory-based machines, you are asked to confirm any FEP command that would prevent you from using the Continue FEP command. On 3600-family machines, if you have used the Halt Machine command or `h-c-FUNCTION` to stop Lisp, loaded new microcode, or used the Clear Machine command, the Continue command will not work. Instead, you'll need to use the Start command (to warm or cold boot). See the section "Using the Genera Application on a MacIvory".

Load the Continue FEP command by scanning the overlay (flod) file `*-lisp.flod`. See the section "Start FEP Command".

Debug FEP Command

Debug

Enters the FEP Debugger on 3600-family machines or the IFEP Debugger on

Ivory-based machines. You can use this command to gather information for sending bug reports. See the section "Debugging in the FEP".

On Ivory-based machines, use the Mail Bug Report subcommand of the Debug FEP Command to append the stack backtrace to the crash data. Crash data for both the 3600 and Ivory machines can be recovered the next time Lisp is running: Use the Command Processor (CP) Show Crash Data command.

Load the Debug FEP command by scanning the overlay (flod) file `*-debug.flod`.

keywords :Ignore Storage Structures, :Show Initial Frame

These keywords are only supported on Ivory-based machines.

:Ignore Storage Structures
{Yes, No} This option allows you to ignore Lisp's storage structure when you invoke the IFEP Debugger.

Specify :Ignore Storage Structures Yes, if you suspect that the Lisp storage system is damaged or is preventing the IFEP Debugger from working properly. This option defaults to Yes if the FEP believes Lisp has not completed initializing its storage system (otherwise, the default is No).

Note that if you specify :Debug :Ignore Storage Structures when Lisp has been running for some time, you may get unpredictable results. For more details, see the section "The IFEP Debugger and Virtual Memory".

:Show Initial Frame {Yes, No} This option allows you to control the display of the initial frame when you invoke the IFEP Debugger. This keyword option defaults to Yes if the FEP believes that Lisp is running (to No, otherwise).

Specify :Debug :Show Initial Frame No if an error in the initial display appears to be preventing you from entering the Debugger.

Declare More Paging Files FEP Command

This command is implemented on both Symbolics 3600-family and Symbolics Ivory-based machines. Although Ivory-based machines do not use hyphens in their FEP command names, many 3600-family machine FEP commands do. Where there are differences, they are shown here.

Declare More Paging-Files *sequence-of-filenames* (for 3600-family machines)

Declare More Paging Files *sequence-of-filenames* (for Ivory-based machines)

Declares *sequence-of-filenames* to be paging files.

sequence-of-filenames
: A list of files separated by spaces on 3600-family machines, and by commas on Ivory-based machines.

Note: The syntax of FEP commands that take sequence arguments is different, depending on whether the machine is a 3600-family machine or an Ivory-based machine. A "sequence argument" is a list of things, such as the *sequence-of-filenames* argument to the Declare Paging Files and the Declare More Paging Files commands.

On 3600-family machines, the elements of the sequence must be separated with spaces. For example:

```
Declare Paging-Files file1 file2 file3
```

On Ivory-based machines, the elements of the sequence must be separated with commas and no spaces. (This syntax is compatible with the Command Processor syntax.) For example:

```
Declare Paging Files file1,file2,file3
```

The Declare More Paging Files FEP command is similar to the Declare Paging Files command, but Declare More Paging Files does not clear previously declared paging files. Instead, it declares new paging files (in addition to any paging files that have already been declared). See the section "Declare Paging Files FEP Command". The Declare More Paging Files FEP command checks to see if each paging file actually exists. If it does not exist, a warning is issued, and the file is added to the list of declared files in case the file is created later.

The list of declared paging files is cleared when:

- You reset the FEP.
- Your machine is powered down.
- You reissue the Declare Paging Files command.

Here is a sample (Ivory-based machine) `hello.boot` file that includes both the Declare Paging Files and the Declare More Paging Files FEP commands:

```
Hello Innn
Hello Local (or hostname)
```

I*nnn* and Local represent two `.boot` files. Their contents should be as follows:

Hello I*nnn* Boot File

The `I`*nnn*`.boot` file (where *nnn* is the IFep version number, which is 325 for Genera 8.1) should contain the commands to scan the flod files and initialize things.

```
Scan I325-lisp.flod
Scan I325-loaders.flod
Scan I325-info.flod
Scan I325-debug.flod
Initialize Hardware Tables
```

Hello Local Boot File

The `local.boot` file should contain those commands that set up this specific machine, declaring paging files, setting the network address, and any other boot options.

```
Declare Paging Files FEP0:>Paging-1.page
Declare More Paging Files FEP0:>Paging-2.page,Paging-3.page
Set Boot Options :Network Address Chaos|52525 :IDS Enable
```

Load the Declare More Paging Files FEP command by scanning the overlay (flod) file `*-loaders.flod`. This command commonly appears in files of the type `boot.boot`, `hello.boot`, and `autoboot.boot`.

Note: Issue the Declare More Paging Files FEP command before the Load World FEP command and after the Declare Paging Files FEP command inside a boot.boot file. Issue the Declare Paging Files and Declare More Paging Files FEP commands after the Initialize Hardware Tables FEP command inside a hello.boot file.

Declare Paging Files FEP Command

This command is implemented on both Symbolics 3600-family and Symbolics Ivory-based machines. Although Ivory-based machines do not use hyphens in their FEP command names, many 3600-family machine FEP commands do. Where there are differences, they are shown here.

Declare Paging-Files *sequence-of-filenames* (for 3600-family machines)

Declare Paging Files *sequence-of-filenames* (for Ivory-based machines)

Declares *sequence-of-filenames* to be paging files.

sequence-of-filenames
: A list of files separated by spaces on 3600-family machines, and by commas on Ivory-based machines.

Note: The syntax of FEP commands that take sequence arguments is different, depending on whether the machine is a 3600-family machine or an Ivory-based machine. A "sequence argument" is a list of things, such as the *sequence-of-filenames* argument to the Declare Paging Files and the Declare More Paging Files commands.

On 3600-family machines, the elements of the sequence must be separated with spaces. For example:

```
Declare Paging-Files file1 file2 file3
```

On Ivory-based machines, the elements of the sequence must be separated with commas and no spaces. (This syntax is compatible with the Command Processor syntax.) For example:

```
Declare Paging Files file1,file2,file3
```

The Declare Paging Files command is similar to the Declare More Paging Files FEP command, but clears any previously declared paging files. See the section "Declare More Paging Files FEP Command". The Declare Paging Files FEP command checks to see if each paging file actually exists. If it does not exist, a warning is issued, and the file is added to the list of declared files in case the file is created later.

If no declared paging files exist, the Load World command attempts to add the paging file called FEP*n*:>Page.page. The list of declared paging files is cleared when:

- You reset the FEP.
- Your machine is powered down.
- You reissue the Declare Paging Files command.

Here is a sample hello.boot file for an Ivory based machine. It contains both the Declare Paging Files and the Declare More Paging Files FEP commands.

```
Hello Innn
Hello Local (or hostname)
```

I*nnn* and Local represent two .boot files. Their contents should be as follows:

Hello I*nnn* Boot File

The I*nnn*.boot file (where *nnn* is the IFep version number, which is 325 for Genera 8.1) should contain the commands to scan the flod files and initialize things.

```
Scan I325-lisp.flod
Scan I325-loaders.flod
Scan I325-info.flod
Scan I325-debug.flod
Initialize Hardware Tables
```

Hello Local Boot File

The local.boot file should contain those commands that set up this specific machine, declaring paging files, setting the network address, and any other boot options.

```
Declare Paging Files FEP0:>Paging-1.page
Declare More Paging Files FEP0:>Paging-2.page,Paging-3.page
Set Boot Options :Network Address Chaos|52525 :IDS Enable
```

Load the Declare Paging Files FEP command by scanning the overlay (flod) file `*-loaders.flod`. This command commonly appears in files of the type `boot.boot`, `hello.boot`, and `autoboot.boot`.

Note: Issue the Declare Paging Files FEP command before the Load World FEP command inside a boot.boot file. Issue the Declare Paging Files FEP command after the Initialize Hardware Tables FEP command inside a hello.boot file.

Enable IDS FEP Command

Enable IDS

Enables the Incremental Disk Save facility. See the section "Using the Incremental Disk Save (IDS) Facility".

See the section "Set Boot Options FEP Command".

Load the Enable IDS FEP command by scanning the overlay (flod) file `*-loaders.flod`. This command commonly appears in files of the type `boot.boot` and `autoboot.boot`.

Note: Issue the Enable IDS FEP command after the Load World FEP command and before the Start FEP command inside a boot file.

Hello FEP Command

Hello *filename*

Loads the hello.boot file.

filename A filename (defaulting to FEP*n*:>hello.boot) that contains a minimum sequence of commands to:

- Scan the FEP overlay (flod) files.
- Determine the machine's hardware configuration.
- Set the machine's primary network address.

Optionally, the hello.boot file can contain additional commands. For more information:

- See the section "Scanning the Overlay (Flod) Files".

- See the section "Declare Paging Files FEP Command".
- See the section "Declare More Paging Files FEP Command".

For Ivory-based machines, also see the section "Set Boot Options FEP Command".

A hello.boot file resides on every Symbolics machine. Each time a machine is powered up or reset, use the Hello command to load the hello.boot file. (On MacIvory machines, this is automatically done for the user whenever Lisp is started). See the section "Using the Genera Application on a MacIvory".

The Hello FEP command is resident in the FEP, so it needn't be loaded from an overlay (flod) file. See the section "Overlay (Flod) Files and the FEP".

Initialize Hardware Tables FEP Command

Initialize Hardware Tables

initializes the FEP's hardware tables.

The Initialize Hardware Tables FEP command determines the type and size of memory on each memory board in the machine. Before this command is executed for the first time, the FEP has no information about the type — or the respective locations — of each memory board in the machine.

It's not necessary to explicitly issue the Initialize Hardware Tables FEP command, since any command that requires hardware table initialization automatically executes that function. Because the initialization function might have to print diagnostic messages to the console, however, it is useful to place Initialize Hardware Tables FEP command in the hello.boot file.

Load the Initialize Hardware Tables FEP command by scanning the overlay (flod) files `*-loaders.flod`. This command commonly appears in files of the type `hello.boot` and `autoboot.boot`.

Load Microcode FEP Command

Note: This command is implemented only on Symbolics 3600-family machines.

Load Microcode *filename*

Loads microcode from the local disk into the machine's memory. You must use the Load Microcode command before loading a world, and also if you want to use the Disk Format or Disk Restore FEP commands. The Clear Machine FEP command should always precede the Load Microcode FEP command.

filename A filename indicating where the microcode is stored. The initial default *filename* for this command depends on the local machine's hardware configuration. For example:

Hardware Configuration	*Default*
3600, FPA	FEP:>3600-fpa-mic.mic
3600, FPA, XSQ	FEP:>3600-fpa-xsq-mic.mic
3640, FPA	FEP:>3640-fpa-mic.mic
3640, FPA, XSQ	FEP:>3640-fpa-xsq-mic.mic

To recompute the initial microcode default, use the Compute Microcode Default command. See the section "Compute Microcode Default FEP Command".

Load the Load Microcode FEP command by scanning the overlay (flod) file `*-loaders.flod`. This command commonly appears in files of the type `boot.boot` and `autoboot.boot`.

Load World FEP Command

Load World *filename*

Loads the Lisp world from a local disk into the machine's memory, and adds any paging files declared with the Declare Paging Files or Declare More Paging Files FEP commands.

filename A filename indicating where the world is stored. The default value of *filename* is the last file name given to the Load World command.

Because the Load World FEP command searches the local machine's (mounted) disks to find any parents of an IDS world loaded with the Load World FEP command, use the Mount FEP command to explicitly mount all disk units (if they are not already mounted). On 3600-family machines, it is possible to netboot using the Load World FEP command.

Note: The Load World FEP command checks the local disks for an IDS world's parents. If one or more parents is missing, the Load World FEP command will look for the parent on all enabled netboot servers, and attempt to netboot it. This means that, if all the parents of an IDS world do not reside on the local disk, Load World becomes a request to netboot the parent worlds of an IDS loaded from the local disk.

Also:

- See the section "Booting IDS Worlds".
- See the section "Netbooting".

Load the Load World FEP command by scanning the overlay (flod) file `*-loaders.flod`. This command commonly appears in files of the type `boot.boot`, and `autoboot.boot`.

Note: The Load World FEP command sets the default disk unit to be the disk from which the world was loaded. To change this default, use the Set Default-Disk-Unit FEP command.

Mount FEP Command

Mount *device*

Mounts a specified device.

device A number indicating which device to mount. For example, to mount disk unit `FEP2:`

```
Mount (default is FEP:) 2
```

Any time a pathname references a device, that device is automatically mounted for you. You must explicitly mount a device only if you haven't used any pathnames that referenced it. See the section "Dismount FEP Command".

The Mount FEP command is resident in the FEP, so it needn't be loaded from an overlay (flod) file.

Netboot FEP Command

Netboot *world-description*

Loads a netboot core from the local disk into the machine's memory, and adds any paging files declared with the Declare Paging Files or Declare More Paging Files FEP commands. This prepares the machine to boot a Lisp world — described by *world-description* — from a netboot server's disk.

world-description Describes which world to boot.

world-description can be a complete specification, or a short, unique substring from the name of a world-load file. *world-description* need not include a disk unit specification, or the file type `.load` or `.ilod`.

If you have two worlds with identical substrings in them, the Netboot FEP command looks for the newest one. For example, if these two worlds exist:

Genera-8-0-incremental-1.load

Genera-8-0-incremental-2.load

and you issue this command:

```
Command: Netboot Genera-8-0
```

you get the newest world containing the substring Genera 8.0 (but you are not able to specify which one you want to boot).

To ensure that you netboot the world you intend, name the worlds something like this:

Inc-1

Inc-2

Then, issue the Netboot FEP command, followed by the appropriate (unique) substring.

For more information about netboot cores, see the section "Netboot Cores". For more information about the Declare Paging Files and Declare More Paging-Files FEP commands, see the section "Declare Paging Files FEP Command" and see the section "Declare More Paging Files FEP Command".

Load the Netboot FEP command by scanning the overlay (flod) file `*-loaders.flod`. This command commonly appears in files of the type `boot.boot`, and `autoboot.boot`.

Note: When you netboot, replace the Load World FEP command with the Netboot FEP command inside a boot file (or manually type the Netboot FEP command within the appropriate sequence). For more information about netbooting, see the section "Netbooting".

Reset FEP FEP Command

Reset FEP

Initializes the FEP's memory. After the FEP is reset, you must initialize the FEP overlay (flod) files by typing Hello to the FEP command prompt. (On MacIvory machines, this is automatically done for the user whenever Lisp is started).

The Reset FEP command is resident in the FEP, so it needn't be loaded from an overlay (flod) file.

Scan FEP Command

Scan *pathname*

Reads an overlay (flod) file, and makes the commands located in the file available to the FEP.

pathname Specifies the overlay (flod) file to read.

The Scan FEP command is resident in the FEP, so it needn't be loaded from an overlay (flod) file. This command commonly appears in files of the type `hello.boot` and `autoboot.boot`.

Set Boot Options FEP Command

Note: This command is implemented only on Symbolics Ivory-based machines.

Set Boot Options *keywords*

Sets default values for *keywords* on the local Ivory-based machine. Use this command instead of entering the corresponding FEP command for each *keyword* in a `hello.boot` file.

The Set Boot Options FEP command makes it possible for users to boot distribution worlds without having to site-configure them first.

*keyword*s :Network Address, :Ethernet Address, :LMFS FSPT Unit, :Timezone Offset, :Timezone Name, :Site Name, :Namespace Descriptor File, :Default World, :Default Boot File, :IDS, :Autoboot on Halt, :Slave-Buffer Base

:Autoboot on Halt
{Yes, No}. If Yes, and Lisp halts to the FEP, the FEP searches for an `autoboot.boot` file and processes it (if found) as if the system had just been powered up. If the autoboot process is aborted (by typing any character before it is finished) this option is set back to No (on the assumption that the system is no longer running unattended).

Recommended usage: Your autoboot.boot file might look like this:

```
Hello
Set Boot Options :Autoboot on Halt Yes
(Type any character to abort autobooting...)
Autoboot Delay 20
Boot
```

Note that the Set Boot Options command must follow the Hello command, since it is in an overlay and would otherwise be unrecognized. (A "safer" `.boot` file would also specify the arguments to Hello and Boot.)

If you abort an autoboot, and later want to leave the machine unattended, you must reset the :Autoboot on Halt keyword. You could have it always set on in your standard hello file or boot file. Another way to do this is simply to halt to the FEP and reboot by:

```
Hello autoboot.boot
```

when you are ready to return to unattended operation. (Note that the default for Hello is always hello.boot, so this is safe to use with the above autoboot file where the Hello command reads its default. If you were to use Boot, instead of Hello above, you would end up recursively reading the autoboot.boot

file. For safety, you should always fully specify arguments to commands in `.boot` files.)

:Default Boot File
Pathname for a boot file to use (must be in the local machine's FEPFS).

:Default World A world to load (must be in the local machine's FEPFS).

:Ethernet Address
The Ethernet address of the local machine. (This is used for DNA only.)

:IDS {Enable, Disable, Default}. How to set IDS when booting. The default is Default, that is, based on the world.

:LMFS FSPT Unit
The disk unit on which the File System Partition Table resides.

:Namespace Descriptor File
Pathname for a locally accessible namespace descriptor file (can be resident in the FEPFS, LMFS, or host (for example, Macintosh) file systems). If a site name is provided and it is not the current site, the site will be changed. If no descriptor file is provided, the site information is obtained from the network.

:Network Address
The primary network address of the local machine. A Chaos address is specified as follows: `CHAOS|24623`. An Internet address is specified as follows: `INTERNET|128.81.41.147`.

:Timezone Name Name of the local machine's timezone.

:Timezone Offset Offset from GMT of the local machine's timezone.

:Site Name Site name for a site. Used to change sites at boot time.

:Slave-Buffer Base
Enables you to specify the base-address when it is something other than the default.

Load the Set Boot Options FEP command by scanning the overlay (flod) file `*-lisp.flod`. This command commonly appears in files of the type `hello.boot`.

For related information, see the section "Set Ethernet Address FEP Command", and see the section "Set Chaos Address FEP Command".

Set Chaos Address FEP Command

Note: This command is implemented only on Symbolics 3600-family machines.

Set Chaos-Address *octal-value*

Sets the Chaos address of the local 3600-family machine. Ivory-based machines must use the Set Network Address FEP command, instead. For more information, see the section "Set Network Address FEP Command".

octal-value Specifies the Chaos address; this defaults to the previously set Chaos address, and to zero when the FEP is reset.

Load the Set Chaos Address FEP command by scanning the overlay (flod) file `*-lisp.flod`. This command commonly appears in files of the type `boot.boot` and `autoboot.boot`.

For related information, see the section "Set Ethernet Address FEP Command", and see the section "Set Network Address FEP Command".

Set Default Disk Unit FEP Command

Note: This command is implemented only on Symbolics 3600-family machines.

Set Default-Disk-Unit *unit-number*

Sets the disk unit to which Lisp and the FEP should default for all subsequent disk references.

unit-number Specifies a unit; must be a number in base 10.

Load the Set Default Disk Unit FEP command by scanning the overlay (flod) file `*-loaders.flod`. This command commonly appears in files of the type `boot.boot`, and `autoboot.boot`.

Note: The Load World FEP command sets the default disk unit to be the disk from which the world was loaded. To change this default, use the Set Default-Disk-Unit FEP command.

Set Display-string FEP Command

Note: This command is supported only on the following machine models: 3600, 3640, 3645, 3670, 3675.

Set Display-string *string*

Displays a string on the front panel of some Symbolics 3600-family machines. This is useful when you want to display the machine's name or Chaos address there, for example.

string The string to display. Its length is limited to 12 characters. If more characters are used, the string is truncated.

Load the Set Display-string FEP command by scanning the overlay (flod) file `*-lisp.flod`. This command commonly appears in files of the type `hello.boot` and `autoboot.boot`.

Set Ethernet Address FEP Command

This command is implemented on both Symbolics 3600-family and Symbolics Ivory-based machines. Although Ivory-based machines do not use hyphens in their FEP command names, many 3600-family machine FEP commands do. Where there are differences, they are shown here.

Set Ethernet-Address *Ethernet-address*

Set Ethernet Address *Ethernet-address*

Sets the Ethernet address of the local machine. Use this command if you are using Symbolics DNA (DECnet).

Ethernet-address The Ethernet address of the local machine.

Load the Set Ethernet Address FEP command by scanning the overlay (flod) file `*-lisp.flod`. This command commonly appears in files of the type `boot.boot` and `autoboot.boot`.

For related information, see the section "Set Network Address FEP Command", see the section "Set Chaos Address FEP Command", and see the section "Show Ethernet Address FEP Command".

Set LMFS FSPT Unit FEP Command

Set LMFS FSPT Unit *unit-number*

In order to use the local Lisp Machine File System (LMFS), Lisp must have access to the File System Partition Table (FSPT). The File System Partition Table is contained within a FEP file named `fspt.fspt` that lists the LMFS partitions. See the section "LMFS Multiple Partitions".

Sets the default location for the file named `fspt.fspt` to the disk unit specified by *unit-number*.

unit-number A number indicating a unit; must be a number in base 10.

Note: The Load World FEP command sets the default disk unit to be the disk from which the world was loaded. To change this default for the FSPT file, use the Set LMFS FSPT Unit FEP command.

Load the Set LMFS FSPT Unit FEP command by scanning the overlay (flod) file `*-lisp.flod`. This command commonly appears in files of the type `hello.boot` and `autoboot.boot`.

Set Network Address FEP Command

This command is implemented on both Symbolics 3600-family and Symbolics Ivory-based machines. Although Ivory-based machines do not use hyphens in their FEP

command names, many 3600-family machine FEP commands do. Where there are differences, they are shown here.

Set Network-Address *interfacename*:*network-type*|*network-address*;*option*;
option;... (for 3600-family machines)

Set Network Address *interfacename*:*network-type*|*network-address*;*option*;
option;... (for Ivory-based machines)

Sets the primary network type and address of the local machine. This command supports the ability to use the Internet as the primary network.

interfacename The name of the interface, if there are multiple network interfaces available. Separate this from *network-type* by a colon [:].

- On 3600-family machines, the interface name is OBS-0 for 3600, 3670, 3675, 3640 and 3645 machines, and NBS-0 for 3620, 3630, and 3650 machines.
- On XL400 machines, the interface name is `Merlin0`.
- On MacIvory machines, the interface name is `enet`*portnumber*.
- On UX-family machines, the interface name is the name of the Sun interface.

network-type Chaos or Internet. Separate this from *network-address* with a bar [|].

network-address The primary network address of the local machine, in the address format appropriate to the network type.

option Additional address information for use on machines that boot in an Internet-only configuration on a network with several subnets. Separate these by semicolons [;]. The two options are:

gateway — The address of the gateway machine on the subnet. This allows the machine to locate its namespace server if the namespace server is on another subnet.

mask — A representation of bits in the network address that should be masked when determining the host number. For an explanation of internet subnet masks, see the section "IP/TCP Support for Subnetting".

If the machine is on more than one network, separate each network address string with commas [,].

The network addresses of a UX-family machine might look like this:

```
ie0:Chaos|24407, ie0:Internet|128.81.41.7;gateway=128.81.41.1;mask=255.255.255.0
```

On Ivory-based machines, load the Set Network Address FEP command by scanning the overlay (flod) file `*-lisp.flod`. On 3600-family machines, load the Set Network-Address FEP command by scanning the overlay (flod) file `*-rel7.flod`. This command commonly appears in files of the type `boot.boot` and `autoboot.boot`.

Note that on Ivory-based machines you can use the Set Boot Options command (specifically, the :Network Address keyword) to set the network address, instead of using the Set Network Address command. See the section "Set Boot Options FEP Command".

For related information, see the section "Set Ethernet Address FEP Command", and see the section "Set Chaos Address FEP Command".

Set Prompt FEP Command

Set Prompt *string*

Sets the FEP command prompt to the specified *string*.

string The string to be used as the FEP command prompt.

For example:

```
FEP Command: Set Prompt "In the FEP Again?"

In the FEP Again?
```

The Set Prompt FEP command is resident in the FEP, so it needn't be loaded from an overlay (flod) file. This command commonly appears in files of the type `hello.boot` and `autoboot.boot`.

Show Directory FEP Command

Show Directory *directory-specification*

Displays the contents of a specified directory in the FEP file system, and shows detailed information about each FEP file system directory entry, similar to the way the CP command Show Directory shows this information.

directory-specification

Specifies a directory in the FEP file system. *directory-specification* can use simple wildcards, as shown in these examples:

Wildcard Spec	*Lists all*
`*.load`	world loads
`*.boot`	boot files
`v127*.flod`	.flod files for FEP version 127
`*sys*.*`	files whose name contains `sys`
`*.mic.428`	version 428 microcode files

The Show Directory FEP command displays:

- The number of blocks allocated to each FEP file.
- The number of bytes in each file and the file's byte-size (or the word "directory" if the file is a subdirectory of FEP*n*).
- Any flags (such as "don't delete", "deleted", and "don't reap").
- The file's creation time (in Greenwich Mean Time (GMT)).
- Comments about the file (if any exist).
- The file's author.

The Show Directory FEP command is resident in the FEP, so it needn't be loaded from an overlay (flod) file.

Show File FEP Command

Show File *filename*

Displays the contents of a file in the the FEP file system.

filename A file in the FEP file system.

The Show File FEP command is resident in the FEP, so it needn't be loaded from an overlay (flod) file.

Show Paging Files FEP Command

This command is implemented on both Symbolics 3600-family and Symbolics Ivory-based machines. Although Ivory-based machines do not use hyphens in their FEP command names, many 3600-family machine FEP commands do. Where there are differences, they are shown here.

Show Paging-Files (for 3600-family machines)

Show Paging Files (for Ivory-based machines)

Shows two lists of files:

1. Files declared using the Declare Paging Files or Declare More Paging Files commands.

2. Files added using the Command Processor (CP) Add Paging File command, or added automatically by the Load World or Netboot FEP command.

Load the Show Paging Files FEP command by scanning the overlay (flod) file `*-loaders.flod`.

Shutdown FEP Command

Shutdown

Asks for confirmation and halts the FEP. On some machine models, the Shutdown command queries appropriately and powers down the machine. On embedded systems, the Shutdown command closes the Genera application and returns control to the embedding host.

To restart (and reset) the machine, press the RESET button on the processor's front panel.

If you have a MacIvory, use the Cold Boot Lisp Ivory menu item.

If you have a UX-family machine, run the `Genera` program.

Use the Shutdown command to power off an XL400 or a 3600-family machine as follows:

1. Issue the Halt Machine command.

2. Issue the Shutdown command.

3. Power off the console and processor.

Use the Shutdown command to power off a MacIvory machine as follows:

1. Use the Shutdown Ivory menu item.

2. Use the Shutdown Special menu item.

3. Power off the console, processor, and external disk, if one exists.

Use the Shutdown command to power off a UX-family machine as follows:

1. Issue the Halt Machine command.

2. Issue the Shutdown command.

3. Shut down UNIX (see your UNIX documentation).

4. Power off the console and processor.

The Shutdown FEP command is resident in the FEP, so it needn't be loaded from an overlay (flod) file.

Start FEP Command

Start *keyword*

Transfers control to the loaded Lisp world.

keyword :Ignore Saved State

:Ignore Saved State
{Yes, No} This keyword argument should not normally need to be used. Yes causes the machine to attempt not to unwind the process out of which you are warm booting. The default is No. This keyword is available only on Ivory-based machines.

If the Lisp world has been running, the Start FEP command initiates a warm boot. If the Lisp world has just been loaded (using the Load World or Netboot command), the Start FEP command initiates a cold boot. See the section "Using the Genera Application on a MacIvory". See the section "Continue FEP Command".

Load the Start FEP command by scanning the overlay (flod) file `*-lisp.flod`.

Color Systems FEP Commands

These FEP commands pertain to systems with color consoles. For your convenience, we have arranged these commands in alphabetical order.

Attach Graphics Tablet FEP Command

Note: This command is implemented only on Symbolics 3600-family machines.

Attach Graphics Tablet *serial-port-number*

Issue this command before connecting a graphics tablet to a serial port.

serial-port-number A number indicating a serial port.

Note: This command is supported only on the following machine models: 3600, 3640, 3645, 3670, 3675.

Load the Attach Graphics Tablet FEP command by scanning the overlay (flod) file `*-lisp.flod`.

Clear Color Background Screen FEP Command

Note: This command is implemented only on Symbolics 3600-family machines.

Clear Color Background Screen

Clears the regular (not overlay) screen in a color console system. Since the FEP writes to the overlay screen, it may be easier to read the overlay screen after you clear the regular screen.

The Clear Color Background Screen FEP command is resident in V127 and G206 versions of the FEP EPROM, so it needn't be loaded from an overlay (flod) file.

Color FEP Command

Note: This command is implemented only on Symbolics 3600-family machines.

Color

Like the Monochrome FEP command, allows you to select between the color and monochrome displays. You must either warm boot or cold boot after issuing this command.

The Color FEP command is resident in V127 and G206 versions of the FEP EPROM, so it needn't be loaded from an overlay (flod) file.

Detach Graphics Tablet FEP Command

Note: This command is implemented only on Symbolics 3600-family machines.

Detach Graphics Tablet

Issue the Detach Graphics Tablet FEP command before disconnecting a graphics tablet from a serial port.

Note: This command is supported only on the following machine models: 3600, 3640, 3645, 3670, 3675.

Load the Detach Graphics Tablet FEP command by scanning the overlay (flod) file `*-lisp.flod`.

Load Color Sync Program FEP Command

Note: This command is implemented only on Symbolics 3600-family machines.

Load Color Sync-Program

Loads a color sync program from disk.

Load the Load Color Sync Program FEP command by scanning the overlay (flod) file `*-loaders.flod`.

Monochrome FEP Command

Note: This command is implemented only on Symbolics 3600-family machines.

Monochrome

LIke the Color FEP command, allows you to select between the monochrome and color displays. You must either warm boot or cold boot after issuing this command.

The Monochrome FEP command is resident in V127 and G206 versions of the FEP EPROM, so it needn't be loaded from an overlay (flod) file.

Set Color Monitor Type FEP Command

Note: This command is implemented only on Symbolics 3600-family machines.

Set Color Monitor-Type *console-name*

Identifies the color monitor to the FEP. This command also loads the appropriate sync program from the PROM and sets the sync program in the CadBuffer (or CadBuffer2) hardware.

console-name Amtron, Tektronix, or Sony

The Set Color Monitor Type FEP command is resident in all versions of the FEP, so it needn't be loaded from an overlay (flod) file.

Set Console FEP Command

Set Console *console keywords*

Allows you to select between the available consoles. For 3600-family machines with CadBuffer2 hardware, this command also switches input from the console's display hardware to the CadBuffer2 keyboard. To make Lisp notice that the console has been changed, you need to reload microcode (on 3600-family machines) and warm or cold boot (on both 3600-family and Ivory machines) after using the Set Console FEP command.

console On 3600-family machines, the *console* is Color or Monochrome. On Ivory-based machines, the *console* can be any of the available and enabled consoles; press `HELP` for a list of choices.

keywords :Clear Screen, :Cold Load Too

:Clear Screen {Yes, No} Whether to clear the screen before selecting it. The default is No.

:Cold Load Too {Yes, No} Whether Lisp's cold-load stream (and Main console if you warm boot) should be redirected there also. The default is Yes.

The Set Console FEP command is resident in G208 and greater versions of the FEP EPROM, and in I322 or greater versions of the IFEP kernel, so it needn't be loaded from an overlay (flod) file.

Set Disk Label FEP Command

Set Disk Label *unit-number keywords*

unit-number A disk unit (must be a number in base 10).

keywords :Color System Startup File, :FEP Kernel, :Query

:Color System Startup File
: Pathname of the file where the color system startup programs are stored.

:FEP Kernel Pathname of the file where the FEP kernel is stored.

:Query {Yes, No} Whether the system should ask for confirmation before setting the disk label. The default is No.

The Set Disk Label command is resident in the FEP, so it needn't be loaded from an overlay (flod) file.

Set FEP Options FEP Command

Set FEP Options *keywords*

Sets options which are used by the FEP.

keywords :Color System Number, :Color System Startup Program, :Color System Type, :Serial Console Type

The Color System keywords are used only when you are using a custom color monitor (other than the default Sony monitor).

:Color System Type
: {None, FrameThrower} FrameThrower enables the FEP to use the color monitor. None disables the use of the color monitor.

:Color System Number
: The number of the color system, a decimal integer between 0 and 255. (You can have more than one FrameThrower, and the color system number enables you to distinguish among them.)

:Color System Startup Program
: Which color system startup program in the color system startup file for the color system to use.

:Serial Console Type
: {None, ASCII, or X3.64} None prevents the FEP from ever us-

ing the serial console; this is appropriate if you have a serial device other than a console connected to the serial port, and know you won't use the serial device as a console. ASCII indicates that the serial console is a dumb terminal. X3.64 indicates that the serial console is an ANSI-standard X3.64 terminal, such as a VT100.

The new options you specify in Set FEP Options take effect when you next reset the FEP with the Reset FEP FEP command.

The Set FEP Options FEP command is resident in the FEP, so it needn't be loaded from an overlay (flod) file.

Set Monitor Type FEP Command

Set Monitor Type *console type* (for 3600-family machines with G208 or greater FEP EPROMs, and for Ivory machines with I322 or greater FEP kernels)

Set Monitor-Type *console-name* (for machines with V127 and G206 FEP EPROMs, and for Ivory machines with I321 or less FEP kernels)

Users of 3600-family machines should use the Set Monitor Type FEP command when installing a monitor that differs from the one specified by the machine's hardware.

On 3600-family machines with G208 or greater FEP EPROMs, sets the console type and name. On 3600-family machines with V127 and G206 FEP EPROMs, sets the console name. This command also loads the appropriate sync program into the machine's display controller, CadBuffer, or CadBuffer2 hardware.

Users of Ivory-based machines should use the Set Monitor Type FEP command if the FEP options installed by the Set FEP Options FEP command are incorrect. At your first opportunity after using the Set Monitor Type command, you should use Set FEP Options to correct the FEP options, and reset the FEP to make the new FEP options take effect.

console — A particular console; this argument defaults to the current console. Use HELP to find out which consoles are applicable. On Ivory machines, a console is specified by three elements: the console type, the sync program, and the unit number. On 3600-family machines, a console is specifed as being color or monochrome

console-name — The list of choices depends on the *console*. Use HELP to find out which choices are applicable.

The Set Monitor Type FEP command is resident in the FEP, so it needn't be loaded from an overlay (flod) file.

Show Disk Label FEP Command

Show Disk Label *unit-number*

Displays the information in the disk label.

unit-number A disk unit (must be a number in base 10).

The Show Disk Label FEP command is resident in the FEP, so it needn't be loaded from an overlay (flod) file.

Systems Maintenance FEP Commands

The FEP commands described in this section are of interest to Site Administrators who maintain systems. For your convenience, we have arranged these commands in alphabetical order.

Add World File

Add World File *pathname*

Adds the world specified by *pathname* into the local machine's internal world database.

Pathname A world load filename (not a netboot *world-description*).

The FEP's internal world database automatically maintains information about the relationships between IDS worlds on the local disk, and is instrumental in determining which world to use as a netboot core.

The Add World File, Clear World Files, Find World Files, and Find All World Files FEP Commands can be used to override the automatically maintained internal world database (for example, if you have multiple copies of the same world on different disks, and you want to force the FEP to find a particular parent for an IDS world).

Here is a sample (3600-family machine) boot file that loads an incremental world with one parent world. This file uses the Add World File FEP command to explicitly name the parent, so that the Load World FEP command won't have to search the local machine's disk to find it. (In this sample file, `genera-7-2.load` is the parent of `incremental-genera-7-2.load`):

```
Clear Machine
Load Microcode FEP1:>3640-mic.mic.430
Add World File FEP0:>genera-8-0.load
Load World FEP1:>incremental-genera-8-0.load
Start
```

For related information,

- See the section "Clear World Files FEP Command".
- See the section "Find World Files FEP Command".
- See the section "Find All World Files FEP Command".
- See the section "Show World Files FEP Command".

Load the Add World File FEP command by scanning the FEP overlay (flod) file `*-loaders.flod`.

Autoboot Delay FEP Command

Autoboot Delay *time-period*

Automatically delays the autobooting process by ten seconds. You can use this command to increase the time period during which you can abort the autobooting process. If you need a longer time period during which to abort autobooting, place this command in your `autoboot.boot` file and specify the the number of seconds you wish to delay autobooting (the number you specify must be a multiple of 10). Placing this command in your `autoboot.boot` file without specifying a number automatically delays autobooting an additional ten seconds.

Note that all commands listed before the Autoboot Delay FEP command in your `autoboot.boot` file are processed before the autoboot delay.

The Autoboot Delay FEP command is available on Ivory-based machines and on 3600-family machines with G208 FEP EPROMS. It is resident in the G208 FEP EPROM, so it need not be loaded from a FEP overlay (flod) file.

Clear Command Tree

Clear Command Tree

Clears FEP memory of all commands that were loaded by scanning the overlay (flod) files.

The Clear Command Tree FEP command is resident in all versions of the FEP, so it needn't be loaded from an overlay (flod) file.

Clear World Files FEP Command

Clear World Files

Removes worlds from the internal world database.

The FEP's internal world database automatically maintains information about the relationships between IDS worlds on the local disk, and is instrumental in determining which world to use as a netboot core.

The Add World File, Clear World Files, Find World Files, and Find All World Files FEP Commands can be used to override the automatically maintained internal world database (for example, if you have multiple copies of the same world on different disks, and you want to force the FEP to find a particular parent for an IDS world).

If you haven't reset the FEP or powered down for several months, the internal world database may contain obselete worlds. Issue the Clear World Files FEP command to clear the database, and free up FEP memory space.

For related information,

- See the section "Add World File FEP Command".
- See the section "Find World Files FEP Command".
- See the section "Find All World Files FEP Command".
- See the section "Show World Files FEP Command".

Load the Clear World Files FEP command by scanning the overlay (flod) file `*-loaders.flod`.

Compute Microcode Default FEP Command

Note: This command is implemented only on Symbolics 3600-family machines.

Compute Microcode Default

Computes the microcode type from the hardware configuration of the local machine, and sets the Load Microcode command so that it will default to the newly computed microcode type. Some examples:

Hardware Configuration	*Microcode Default*
3600, FPA	FEP: 3600-fpa-mic.mic
3600, FPA, XSQ	FEP: 3600-fpa-xsq-mic.mic
3640, FPA	FEP: 3640-fpa-mic.mic
3640, FPA, XSQ	FEP: 3640-fpa-xsq-mic.mic

Load the Compute Microcode Default FEP command by scanning the overlay (flod) file `*-loaders.flod`.

Create Initial FEP Filesystem FEP Command

Note: This command is implemented only on Symbolics Ivory-based machines.

:Create Initial FEP Filesystem *unit keywords*

unit A disk unit number in base 10.

keywords: :Pack-Name, :Creator, :Bad-Blocks, :Creation-Date

:Bad-Blocks {Ask, None, Defect-Data}. Ask queries for a list of bad blocks. None is the default for MacIvory machines. Defect-Data reads the list stored on the disk.

:Creator A string (the author for initial files being created)

:Creation-Date Current date (defaults to the Macintosh computer's clock) in this format:

```
[YYYY-MM-DD HH:MM:SS (Timezone)]
```

:Pack-Name A string that appears in the disk label.

This command is just the part of the Disk Format FEP command that you need for embedded systems. It builds an empty filesystem on a disk for use by Lisp (and the FEP). In standalone systems this command is automatically invoked as a part of Disk Format. In embedded systems, where you are typically allocating only a portion of the host disk for use by Lisp, you do not format the host disk (formatting and partitioning is done by the host operating system and utilities). You still have to create an initial (empty) filesystem in the partition allocated to Ivory before Lisp or the FEP can use it. Use this command to do so.

Warning: This is a dangerous command that can erase all the information on your disk. Don't use this command unless you know how. The typical use of this command is to initialize a new disk that you are adding to your system. It is extremely important to specify the correct unit number; if you specify the wrong unit, the data on that disk will be erased.

The unit number you specify must refer to a disk that does not yet have a filesystem allocated to Ivory. One safe way to figure out which unit number to specify is to try Show Directory on FEP0 (and then on FEP1, and so on). If you get results (files appear in the directory listing), then that unit number refers to an existing filesystem. If, for example, the Show Directory on FEP1 results in an error, then 1 is the unit number that you should specify.

Disable IDS FEP Command

Disable IDS

Disables the Incremental Disk Save facility.

See the section "Using the Incremental Disk Save (IDS) Facility".

See the section "Set Boot Options FEP Command".

Note: Issue the Disable IDS FEP command after the Load World FEP command and before the Start FEP command inside a boot file.

Load the Disable IDS FEP command by scanning the FEP overlay (flod) file `*-loaders.flod`.

Disable Load To Paging Migration FEP Command

This command is implemented on both Symbolics 3600-family and Symbolics Ivory-based machines. Although Ivory-based machines do not use hyphens in their FEP command names, many 3600-family machine FEP commands do. Where there are differences, they are shown here.

Disable Load-to-Paging Migration (for 3600-family machines)

Disable Load to Paging Migration (for Ivory-based machines)

Prevents unmodified world-load file pages from being written out to a paging file. Use this (default) command if you want to allow only modified world-load file pages to be written out to a paging file. See the section "Set Boot Options FEP Command".

Load the Disable Load to Paging Migration FEP command by scanning the overlay (flod) file `*-loaders.flod`.

Dismount FEP Command

Dismount *device*

Dismounts a specified device. This example shows how to dismount disk unit `FEP2:`

```
Dismount (default is FEP:) 2
```

For related information, see the section "Mount FEP Command". The Dismount FEP command is resident in the FEP, so it needn't be loaded from an overlay (flod) file.

Enable Load To Paging Migration FEP Command

This command is implemented on both Symbolics 3600-family and Symbolics Ivory-based machines. Although Ivory-based machines do not use hyphens in their FEP command names, many 3600-family machine FEP commands do. Where there are differences, they are shown here.

Enable Load-to-Paging Migration (for 3600-family machines)

Enable Load to Paging Migration (for Ivory-based machines)

Allows unmodified world-load file pages to be written out to a paging file. Use this command if you want to allow both modified and unmodified world-load file pages to be written out to (and subsequently read from) a paging file. See the section "Set Boot Options FEP Command".

Effective available paging space is increased when you use the Disable Load to Paging Migration FEP command. Load to Paging Migration is disabled by default. For more information about the Disable Load to Paging Migration FEP command, see the section "Disable Load to Paging Migration FEP Command".

Load the Enable Load to Paging Migration FEP command by scanning the overlay (flod) file `*-loaders.flod`.

Note: Issue the Enable Load to Paging Migration FEP command after the Load World FEP command and before the Start FEP command inside a boot file.

Find All World Files FEP Command

Note: This command is implemented only on Symbolics Ivory-based machines.

Find All World Files

Examines world files in all FEP directories, and updates the internal world database to include the worlds that it finds. As each file is found, its name and world-generation are displayed.

The FEP's internal world database automatically maintains information about the relationships between IDS worlds on the local disk, and is instrumental in determining which world to use as a netboot core.

The Add World File, Clear World Files, Find World Files, and Find All World Files FEP Commands can be used to override the automatically maintained internal world database (for example, if you have multiple copies of the same world on different disks, and you want to force the FEP to find a particular parent for an IDS world).

For related information,

- See the section "Add World File FEP Command".
- See the section "Clear World Files FEP Command".
- See the section "Find World Files FEP Command".
- See the section "Show World Files FEP Command".

Load the Find All World Files FEP command by scanning the overlay (flod) file `*-loaders.flod`.

Find World Files FEP Command

Find World Files *pathname*

Examines world files in the specified FEP directory, and updates the internal world database to include the worlds that it finds. As each file is found, its name and world-generation are displayed.

The FEP's internal world database automatically maintains information about the relationships between IDS worlds on the local disk, and is instrumental in determining which world to use as a netboot core.

The Add World File, Clear World Files, Find World Files, and Find All World Files FEP Commands can be used to override the automatically maintained internal world database (for example, if you have multiple copies of the same world on different disks, and you want to force the FEP to find a particular parent for an IDS world).

For related information,

- See the section "Add World File FEP Command".
- See the section "Clear World Files FEP Command".
- See the section "Find All World Files FEP Command".
- See the section "Show World Files FEP Command".

Load the Find World Files FEP command by scanning the overlay (flod) file `*-loaders.flod`.

Load Sync Program FEP Command

Note: This command is implemented only on Symbolics 3600-family machines.

Load Sync-Program *filename*

Loads the specified file (of type `.sync`) into the 3600-family machine's display controller memory. Issue this command when using a monitor that requires a different sync program than the one resident within the FEP. (The sync program converts bits into the signal that's sent to the monitor.)

The default value of *filename* is the last filename given to the Load Sync Program command. Its initial default is `FEP`*n*`:>Sync.sync`, where FEP*n* is the default disk unit.

Load the Load Sync Program FEP command by scanning the overlay (flod) file `*-loaders.flod`.

Reset Device FEP Command

Reset Device *pathname*

Performs a device-dependent reset of the device specified by *pathname*. Use the command like this:

```
FEP Command: Reset Device FEP1:>
```

The Reset Device FEP command is resident in the FEP, so it needn't be loaded from an overlay (flod) file.

Reset Most FEP Command

Note: This command is implemented only on Symbolics 3600-family machines.

Reset Most

Resets most of the machine, including its processor clock, Lbus, sequencer data paths, display controller, and disks. (If the Reset Most FEP command doesn't "unwedge" the local machine, you probably need to try power-cycling it.)

Load the Reset Most FEP command by scanning the overlay (flod) file `*-lisp.flod`.

Reset Video FEP Command

Note: This command is implemented only on Symbolics 3600-family machines.

Reset Video

Reloads the screen's sync program.

The Reset Video FEP command is resident in the FEP, so it needn't be loaded from an overlay (flod) file.

Retension Cartridge Tape FEP Command

Note: This command is implemented only on Symbolics 3600-family machines.

Retension Cartridge-Tape

Rewinds the tape in the local cartridge tape drive.

This command is resident in 3610, 3620, 3630, 3650, and 3653 machine model FEP EPROMs. If you have a 3600, 3640, 3645, 3670, or 3675 machine model, load the Retension Cartridge-Tape FEP command from the `*-disk.flod` overlay (flod) file.

Set Monitor Type FEP Command

Set Monitor Type *console type* (for 3600-family machines with G208 or greater FEP EPROMs, and for Ivory machines with I322 or greater FEP kernels)

Set Monitor-Type *console-name* (for machines with V127 and G206 FEP EPROMs, and for Ivory machines with I321 or less FEP kernels)

Users of 3600-family machines should use the Set Monitor Type FEP command when installing a monitor that differs from the one specified by the machine's hardware.

On 3600-family machines with G208 or greater FEP EPROMs, sets the console type and name. On 3600-family machines with V127 and G206 FEP EPROMs, sets

the console name. This command also loads the appropriate sync program into the machine's display controller, CadBuffer, or CadBuffer2 hardware.

Users of Ivory-based machines should use the Set Monitor Type FEP command if the FEP options installed by the Set FEP Options FEP command are incorrect. At your first opportunity after using the Set Monitor Type command, you should use Set FEP Options to correct the FEP options, and reset the FEP to make the new FEP options take effect.

console — A particular console; this argument defaults to the current console. Use HELP to find out which consoles are applicable. On Ivory machines, a console is specified by three elements: the console type, the sync program, and the unit number. On 3600-family machines, a console is specifed as being color or monochrome

console-name — The list of choices depends on the *console*. Use HELP to find out which choices are applicable.

The Set Monitor Type FEP command is resident in the FEP, so it needn't be loaded from an overlay (flod) file.

Set World To Netboot FEP Command

Note: This command is implemented only on Symbolics 3600-family machines.

Set World-To-Netboot *world-description*

Prepares the machine to netboot the world specified by *world-description*. Before you use the Set World-to-Netboot FEP command, issue the Load World FEP command to explicitly load a world to be used as the netboot core.

The Set World-To-Netboot FEP command enables you to netboot using a different netboot core than the one that the Netboot command would have automatically selected for you.

Use this command if the netboot core that the Netboot FEP command selects for you proves to be damaged, or doesn't work properly.

Load the Set World To Netboot FEP command by scanning the NFEP overlay (flod) file `*-loaders.flod`.

For information about netbooting, see the section "Netbooting". For information about netboot cores, see the section "Netboot Cores".

Show Command Modules FEP Command

Show Command Modules

Displays the overlay (flod) files that contain accessible commands. If the overlay file `FEP0:>v127-loaders.flod` is not currently loaded in your environment, issuing

the Show Command Modules FEP command will display this:

```
Pathname FEP0:>v127-loaders.flod, not loaded
```

Use the Show Version FEP command to determine the FEP (EPROM or software) version with which your machine has been equipped. (For information about the Show Version FEP command, see the section "Show Version FEP Command".)

If you have a Symbolics 3600-family machine, and it is equipped with an EPROM whose version number is lower than 127, please contact Symbolics Customer Service for an upgrade.

Load the Show Command Modules FEP command by scanning the overlay (flod) file `*-info.flod`.

Show Command Tree

Show Command Tree

Displays the commands that are accessible from scanning the overlay (flod) files.

The Show Command Tree FEP command shows whether a command came from an overlay file or resides within the FEP. It also shows the address within FEP memory at which the command resides (or would reside if the overlay file were loaded).

Load the Show Command Tree FEP command by scanning the overlay (flod) file `*-info.flod`.

Show Configuration FEP Command

Show Configuration

Displays the hardware configuration of the local machine. On MacIvory machines, select the About the Finder Apple menu item for additional information (about the Macintosh host).

Load the Show Configuration FEP command by scanning the overlay (flod) file `*-info.flod`.

Show Disk Label FEP Command

Show Disk Label *unit-number*

Displays the information in the disk label.

unit-number A disk unit (must be a number in base 10).

The Show Disk Label FEP command is resident in the FEP, so it needn't be loaded from an overlay (flod) file.

Show Ethernet Address FEP Command

This command is implemented on both Symbolics 3600-family and Symbolics Ivory-based machines. Although Ivory-based machines do not use hyphens in their FEP command names, many 3600-family machine FEP commands do. Where there are differences, they are shown here.

Show Ethernet-Address (for 3600-family machines)

Show Ethernet Address (for Ivory-based machines)

Displays the Ethernet address of the local machine. Use this command if you are using Symbolics DNA (DECnet).

Load the Show Ethernet Address FEP command by scanning the overlay (flod) file `*-info.flod`.

For related information, see the section "Set Network Address FEP Command", see the section "Set Chaos Address FEP Command", and see the section "Set Ethernet Address FEP Command".

Show LMFS FSPT Unit FEP Command

Show LMFS FSPT Unit

In order to use the local Lisp Machine File System (LMFS), Lisp must have access to the File System Partition Table (FSPT). The File System Partition Table is contained within a FEP file named `fspt.fspt` that lists the LMFS partitions.

The Show LMFS FSPT Unit FEP command shows the disk unit on which Lisp expects the file system partition table to reside.

Load the Show LMFS FSPT Unit FEP command by scanning the overlay (flod) file `*-lisp.flod`.

Note: To change the default location for the FSPT file, use the Set LMFS FSPT Unit FEP command. For more information, see the section "Set LMFS FSPT Unit FEP Command".

Show Serial FEP Command

Note: This command is implemented only on Symbolics 3600-family machines.

Show Serial *unit-number*

Tells whether the specified serial port is in use. If the serial port is in use, the Show Serial FEP command provides information about the port's status.

u*nit-number* A serial port (must be a number in base 10).

Note: This command is supported only on the following machine models: 3600, 3640, 3645, 3670, 3675.

Load the Show Serial FEP command by scanning the overlay (flod) file `*-lisp.flod`.

Show Status FEP Command

Show Status

Displays the internal status of the machine (in machine-dependent format). This command is useful for debugging and crash analysis. Additional crash data can be obtained by using the Debug FEP command. On Ivory-based machines, the Debug command can be used to append a backtrace to the crash data. See the section "Debug FEP Command".

For related information (about the Show Status FEP Command's Command Processor equivalent), see the section "Show Crash Data Command".

Information is available to help you interpret the output of this command for 3600-family machines. See the section "Interpreting the Show Status Command's Output on 3600-Family Machines".

Load the Show Status FEP command by scanning the overlay (flod) file `*-lisp.flod`.

Show Version FEP Command

Show Version

Displays the version of the local machine's FEP (Symbolics 3600-family machines display a FEP EPROM version; Symbolics Ivory-based machines display the version number of their FEP kernel code).

The Show Version FEP command is resident in the FEP, so it needn't be loaded from an overlay (flod) file.

Show World Files FEP Command

Show World Files

Lists the files contained in the internal world database. As each file is listed, the following information is displayed:

- The world-file pathname
- The required microcode (on 3600-family machines)
- The world-file's world-generation

- The world-file's timestamp and the timestamps of its parents
- Whether the world-file can be used as a netboot core

The FEP's internal world database automatically maintains information about the relationships between IDS worlds on the local disk, and is instrumental in determining which world to use as a netboot core.

The Add World File, Clear World Files, Find World Files, and Find All World Files FEP Commands can be used to override the automatically maintained internal world database (for example, if you have multiple copies of the same world on different disks, and you want to force the FEP to find a particular parent for an IDS world).

For related information,

- See the section "Add World File FEP Command".
- See the section "Clear World Files FEP Command".
- See the section "Find World Files FEP Command".
- See the section "Find All World Files FEP Command".

Load the Show World Files FEP command by scanning the overlay (flod) file `*-loaders.flod`.

FEP Commands for Systems Experts

To use the FEP commands in this section, you should have extensive knowledge about the Lisp Machine File System. (Many of these commands can be destructive, so use them all with caution). For your convenience, we have arranged these commands in alphabetical order.

Add Disk Type FEP Command

Note: This command is implemented only on Symbolics 3600-family machines.

Add Disk-Type *name cylinders heads sectors gap1 gap2 gap3 fast*

Lets you declare arbitrary disk types to the FEP. Use this command if you need to format and restore disks that Symbolics does not yet support. You can declare up to four disk types before you need to give the Clear Disk Types FEP command. Add Disk Type is needed only to format and restore disks. It is not needed for the normal operation of any validly formatted disk with a FEP file system.

Add Disk Type has the following arguments, for which it prompts with the argument names in parentheses:

name	The textual name by which this disk type is known.
cylinders	The number of cylinders supported by the drive.
heads	The number of heads on the drive.
sectors	The number of sectors.
gap1	The length of "gap1".
gap2	The length of "gap2".
gap3	The length of "gap3".
fast	0 for slower disks, 1 for faster disks.

These numbers require careful computation and involve some hardware restrictions. The calculations should be performed by Symbolics Customer Service personnel.

Load the Add Disk Type FEP command by scanning the overlay (flod) file `*-disk.flod`.

Clear Disk Counters FEP Command

Note: This command is implemented only on Symbolics 3600-family machines.

Clear Disk-Counters

Resets the registers that keep track of disk statistics.

Load the Clear Disk Counters FEP command by scanning the overlay (flod) file `*-disk.flod`.

Clear Disk Types FEP Command

Note: This command is implemented only on Symbolics 3600-family machines.

Clear Disk-Types

Clears all disk types declared with the Add Disk Type FEP command. See the section "Add Disk Type FEP Command".

Load the Clear Disk Types FEP command by scanning the overlay (flod) file `*-disk.flod`.

Copy File FEP Command

Note: This command is implemented only on Symbolics 3600-family machines.

Copy File *from-pathname to-pathname*

Copies a file from disk to tape on that same machine.

from-pathname Pathname of a file on disk.

to-pathname Pathname of the file on tape.

The Copy File FEP command is resident in the FEP, so it needn't be loaded from an overlay (flod) file.

Debug FEP Command

Debug

Enters the FEP Debugger on 3600-family machines or the IFEP Debugger on Ivory-based machines. You can use this command to gather information for sending bug reports. See the section "Debugging in the FEP".

On Ivory-based machines, use the Mail Bug Report subcommand of the Debug FEP Command to append the stack backtrace to the crash data. Crash data for both the 3600 and Ivory machines can be recovered the next time Lisp is running: Use the Command Processor (CP) Show Crash Data command.

Load the Debug FEP command by scanning the overlay (flod) file `*-debug.flod`.

keywords :Ignore Storage Structures, :Show Initial Frame

These keywords are only supported on Ivory-based machines.

:Ignore Storage Structures
{Yes, No} This option allows you to ignore Lisp's storage structure when you invoke the IFEP Debugger.

Specify :Ignore Storage Structures Yes, if you suspect that the Lisp storage system is damaged or is preventing the IFEP Debugger from working properly. This option defaults to Yes if the FEP believes Lisp has not completed initializing its storage system (otherwise, the default is No).

Note that if you specify :Debug :Ignore Storage Structures when Lisp has been running for some time, you may get unpredictable results. For more details, see the section "The IFEP Debugger and Virtual Memory".

:Show Initial Frame {Yes, No} This option allows you to control the display of the initial frame when you invoke the IFEP Debugger. This keyword option defaults to Yes if the FEP believes that Lisp is running (to No, otherwise).

Specify :Debug :Show Initial Frame No if an error in the initial display appears to be preventing you from entering the Debugger.

Disk Format FEP Command

Disk Format (for 3600-family machines)

Disk Format *unit keywords* (for MacIvory machines)

Formats a disk and creates an initial FEPFS on the local machine. This command is primarily for the use of system maintainers in debugging unusual problems.

When the Disk Format FEP command is used on a 3600-family machine, it creates the root-directory, free-pages, and bad-blocks files in the new FEPFS.

Table ! shows valid answers that you can provide when queried by the FEP.

Questions	*Valid Answers*
Of Type	M2284, T306, M2284, M2294, M2312, M2351A, XT1140, XT1105, XT2190, D2257, P807, EMD368, EMD515, XMD858, XT4380
On Unit	Disk Unit Number
With Pack ID	0
From Cylinder	Cylinder Number (includes lower bound)
Through Cylinder	Cylinder Number (includes upper bound)

Table 4. Using the Disk Format FEP Command on a 3600-Family Machine

Issue the Show Disk Label FEP command to find the answers to these questions. For information about the Show Disk Label FEP command, see the section "Show Disk Label FEP Command".

On MacIvory machines with customer-supplied disk drives, the Disk Format FEP command also initializes a disk partition for Ivory to use (an Ivory partition must already exist on the Macintosh disk).

In general, the defaults for the Disk Format FEP command are correct, and you should not have to supply any keywords.

unit	A disk unit number in base 10.
keywords:	:Pack-Name, :Creator, :Bad-Blocks, :Creation-Date
:Pack-Name	A string that appears in the disk label.
:Creator	A string (the author for initial files being created)
:Bad-Blocks	{ask, none, defect-data}. Ask queries for a list of bad blocks. None is the default for MacIvory machines. Defect-Data reads the list stored on the disk.
:Creation-Date	Current date (defaults to the Macintosh computer's clock) in this format:

```
[YYYY-MM-DD HH:MM:SS (Timezone)]
```

Note: Some FEP commands (like Disk Restore and Disk Format) are used primarily by System Administrators to debug unusual problems. Be careful when using these commands. *If you make a mistake, you can destroy the state of the Lisp file system.*

For related information, see the section "Disk Restore FEP Command".

Load the Disk Format FEP command by scanning the overlay (flod) file `*-disk.flod`.

Disk Restore FEP Command

Disk Restore

Loads FEP files from cartridge tape to disk.

Using the Disk Restore FEP Command on 3600-Family Machines

On 3600-family machines, the cartridge tape from which you are loading files must have been written using either the FEP-Tape Activity or the Copy File FEP command. Before using the Disk Restore FEP command on a 3600-family machine, issue the Clear Machine FEP command and then the Load Microcode FEP command (with the cart: argument; note the trailing colon [:] after cart).

A destination file (large enough to accommodate the restored data) must already exist on the 3600-family machine's disk. The destination file (including its header information) will be overwritten when the data is restored to it from tape.

When you issue the Disk Restore FEP command on a 3600-family machine, you are queried:

```
Have you used Set Disk Type for all units that do not have valid label
blocks?
```

Answer Y (for yes) unless you're installing a new disk (you must set the disk type — using the Set Disk Type FEP command — if the label block is not yet written).

The first file's name, length, author, creation date and time, and comments are displayed from tape and you are queried:

```
Do you want to restore it?
```

- Answer Y (for yes) if you want the program to restore the file.
- Answer N (for no) if you do not want the program to restore the file.
- Answer S (for skip microcodes) if you want the program to search the tape for the next, non-microcode file (a world load, for example).

- Answer F (for find microcode) if you want the program to ask for a specific microcode to find (the default is the current microcode).

Using the Disk Restore FEP Command on Ivory-Based Machines

On Ivory-based machines, the cartridge tape from which you are loading files must have been written using the FEP-Tape Activity. Disk Restore for Ivory machines takes two arguments, *unit* and *network-address* followed by optional keywords:

unit	{*integer*} The disk unit to restore.
network-address	{*string*} The primary network address to use in the `Hello.boot` file.
keywords	:Create Boot, :Create Paging Files, :Creation Date, :Query, :Fep Version, :Format, :Paging Space, :Report, :Source, :Standalone Site, :World File
:Create Boot Files	{Yes, No} Whether to create `Hello.boot` and `Boot.boot` files.
:Create Paging Files	{Yes, No} Whether to create the standard paging files (and declare them in the Hello.boot file).
:Creation Date	{*yyyy-mm-dd hh:mm:ss*} The current date and time.
:Query	{Everything, Files, Confirm, No} Whether to ask about every step, each file to be restored, just confirm before starting, or not ask at all.
:Fep Version	{*integer*} (a version number) The version of the FEP to restore.
:Format	{Yes, No} Whether to format and create an initial FEP file system on the disk first.
:Paging Space	{*integer*} Number of blocks to allocate for paging files.
:Report	{Yes, No} Whether to describe progress.
:Source	{*pathname*} Source device or directory to copy flods and world from.
:Standalone Site	{Yes, No} Whether to include standalone site commands in `Hello.boot`.
:World File	{*pathname*} The pathname of the Lisp world to restore.

The defaults are correct for a complete restore of a Disk from Tape. They can be overridden for other purposes, for example, restoring from Disk to Disk, creating paging files. A simple way to pick and choose is simply to use:

```
Disk Restore N network|address :Query Everything
```

And answer y or n to each query.

Note: Some FEP commands (like Disk Restore and Disk Format) are used primarily by System Administrators to debug unusual problems. Be careful when using these commands. *If you make a mistake, you can destroy the state of the Lisp file system.*

For related information, see the section "Disk Format FEP Command".

Load the Disk Restore FEP command by scanning the overlay (flod) files `*-disk.flod`.

Enable Trap Handling FEP Command

Note: This command is implemented only on Symbolics 3600-family machines.

Enable Trap Handling

On 3600, 3640, 3645, 3670, and 3675 machines, enables interrupts for the FEP doorbell (an interaction between Lisp and the FEP), enables DMA interrupts, and initializes (sets trap enable for) a status register.

On other 3600-family machines, the Enable Trap Handling FEP command simply initializes (sets trap enable for) a status register.

Use the Enable Trap Handling FEP command if, by mistake, some software has disabled trap handling and "forgotten" to turn it back on.

Load the Enable Trap Handling FEP command by scanning the overlay (flod) files `*-lisp.flod`.

Load Complete World FEP Command

Note: This command is implemented only on Symbolics 3600-family machines.

Load Complete World

Loads all of a world into memory. This command is mainly for use by system developers.

Load the Load Complete World FEP command by scanning the overlay (flod) files `*-loaders.flod`.

Load FEP FEP Command

Note: This command is implemented only on Symbolics 3600-family machines.

Load FEP *filename*

Loads and starts loadable FEP programs.

filename Specifies the FEP program. The names of the FEP programs are usually of the form V127-*name*, where V127 is the number of the FEP version on which the program runs and *name* is the name of the program.

Use the Show Version FEP command to determine the FEP (EPROM or software) version with which your machine has been equipped. (For information about the Show Version FEP command, see the section "Show Version FEP Command".)

If you have a Symbolics 3600-family machine, and it is equipped with an EPROM whose version number is lower than 127, please contact Symbolics Customer Service for an upgrade.

The Load FEP FEP command is resident in the FEP, so it needn't be loaded from an overlay (flod) file.

Set Disk Type FEP Command

Note: This command is implemented only on Symbolics 3600-family machines.

Set Disk-Type *unit type pack-id*

Tells the FEP that disk *unit* is of type *type* and has pack id *pack-id*. Disk Restore might need this information if the disk has no label block or if the label block contains incorrect information. Issue the Set Disk Type FEP command after using the Mount FEP command.

unit A number specifying a disk unit.

type The type of the disk unit.

pack-id The pack id of the disk unit.

Load the Set Disk Type FEP command by scanning the overlay (flod) file `*-disk.flod`.

Set Lisp Release FEP Command

Set Lisp Release

Sets the intended Lisp release version.

Load the Set Lisp Release FEP command by scanning the overlay (flod) file `*-lisp.flod`.

Set Wired Addresses FEP Command

Set Wired Addresses *%wired-virtual-address-high*

Sets values for wired addresses. If there are local or Symbolics-distributed patches to the wired system, and if these patches cause an internal limit to be exceeded, an error is signalled stating that the variable **sys:%wired-virtual-address-high** needs to be increased to the suggested new value. This command makes it easy to set the necessary variables.

Load the Set Wired Addresses FEP command by scanning the overlay (flod) file `*-loaders.flod`.

Note: This command must be executed after the Load World command and before the Start command.

Show Disk Types FEP Command

Note: This command is implemented only on Symbolics 3600-family machines.

Show Disk Types

Lists the model names for possible disks, along with disk geometry, including format gap sizes.

Load the Show Disk Types FEP command by scanning the FEP overlay (flod) file `*-disk.flod`.

Test A Memory FEP Command

Note: This command is implemented only on Symbolics 3600-family machines.

Test A-memory

Tests all locations in A memory.

Load the Test A Memory FEP command by scanning the overlay (flod) file `*-tests.flod`.

Test All FEP Command

Test All

Runs all FEP tests on a 3600-family machine. On an Ivory machine, it runs only Test Main Memory. For information about each FEP test:

See the section "Test Main Memory FEP Command".
See the section "Test Simple Main Memory FEP Command".
See the section "Test A Memory FEP Command".
See the section "Test Disks FEP Command".

You can load the Test All FEP command by scanning the overlay (flod) file `*-tests.flod`.

Test Location FEP Command

Test location on a 3600-family machine:

Test Location

Tests a single location in main memory.

On an Ivory machine:

Test Location *address keywords*

address {*octal-number*} The location to test, in octal.

keywords :Base, :Hard ECC Error Action, :Passes, :Test Pattern

:Base {2, 8, 10, 16} The base in which to display the errors. The default is 8.

:Hard ECC Error Action
{Halt, Report, Clear} The action to take when a hard ECC error is encountered. The default is halt, so that the error can be fixed. Report means just report the error. Clear means clear the location. Clearing the location can damage a running world. This action should be used with caution.

:Passes {*integer*} The number of times to run each test pattern over the location.

:Test Pattern {Quick, Checkerboard, Walking-bit, All} The pattern to use to test. The default is Checkerboard. All uses each pattern in turn.

You can load the Test Location FEP command by scanning the overlay (flod) file `*-tests.flod`.

Test Disks FEP Command

Note: This command is implemented only on Symbolics 3600-family machines.

Test Disks

Tests the local machine's disks.

Load the Test Disks FEP command by scanning the NFEP overlay (flod) file `*-tests.flod`.

Test Simple Main Memory FEP Command

Test simple main memory on 3600-family and Ivory machines:

Test Simple Main Memory

Runs a memory test. It runs faster (and more thoroughly) than Test Main Memory.

See the section "Test Main Memory FEP Command".

Load the Test Simple Main Memory FEP command by scanning the overlay (flod) file `*-tests.flod`.

Test Main Memory FEP Command

Test main memory on a 3600-family machine:

Test Main Memory

Tests some locations in main memory. It runs more slowly (and less thoroughly) than Test Simple Main Memory.

See the section "Test Simple Main Memory FEP Command".

On an Ivory machine:

Test Main Memory *keywords*

keywords :Base, :Hard ECC Error Action, :Passes, :Test Pattern

:Base {2, 8, 10, 16} The base in which to display the errors. The default is 8.

:Hard ECC Error Action
{Halt, Report, Clear} The action to take when a hard ECC error is encountered. The default is halt, so that the error can be fixed. Report means just report the error. Clear means clear the location. Clearing the location can damage a running world. This action should be used with caution.

:Passes {*integer*} The number of times to run each test pattern over the memory.

:Start {*location*} The address at which to start the test. There is a lower limit of 1011015 (octal) to protect the FEP itself, which on an Ivory machine resides in main memory.

:Test Pattern {Quick, Checkerboard, Walking-bit} The pattern to use to test. The default is Checkerboard.

You can load the Test Main Memory FEP command by scanning the overlay (flod) file `*-tests.flod`.

FEP-Related Command Processor (CP) Commands

These commands can be typed in at the Command Processor (CP) prompt in the Lisp Listener. For your convenience, we have arranged these commands in alphabetical order.

Add Paging File Command

Add Paging File *pathname :prepend*

Enables you to add a paging file at the Command Processor prompt (in Lisp), rather than from within the FEP. If the paging file does not already exist, use the Create FEP File command to create it. See the section "Create FEP File Command".

pathname — The pathname of the existing FEP file, which becomes the new paging file. The default pathname is the disk unit from which you most recently booted. For example, if you most recently booted from FEP1:>, the default paging file might look like:

```
FEP1:>.page
```

Each paging file must have a unique name.

keywords — :Prepend

:Prepend — {Yes, No} Yes puts the added paging file at the beginning of the list of existing paging files. This makes the newly added paging file available for immediate use. No (the default) puts the added paging file at the end of the list of existing paging files, so that it won't be used immediately.

Create FEP File **Command**

Create FEP File *FEP-file-spec size*

Creates a file on a FEP directory on your machine.

FEP-file-spec — The pathname of the file to create. The default is FEP:>temporary.temp.

size — The size in FEP blocks of the file. You must supply this.

Use Create FEP File to do the following:

- To create an extra paging file. For example:

```
Create FEP File fep0:>aux.page 100000
```

- To allocate space into which to load a world load. For example:

```
Create FEP File fep0:>release-8-0.load 50000
```

Copy Flod Files Command

Copy Flod Files *keywords*

Copies FEP overlay (flod) files to a Symbolics 3600-family machine, or flod files and a FEP kernel to to an Ivory-based machine. On Ivory-based machines, Copy Flod Files also makes sure that your FEP kernel and overlay versions are consistent with one another, and installs the previous FEP kernel as the FEP backup kernel. On XL1200 Color Systems, Copy Flod Files also copies the color system startup file. We recommend that you do not delete previous versions of the flod files and FEP kernel, because they can be useful in debugging certain problems.

keywords :Automatic, :Create Hello File, :Disk Unit, :From Directory, :Hosts, :Silent, :Version

:Automatic Whether to automatically skip copying the flod files to any hosts for which the process gets an error. The default is Yes if more than one target host is specified, otherwise No. The mentioned default in both cases is Yes.

:Create Hello File
{Yes, No, Ask} Whether or not to create a `Hello.boot` file after copying (if one does not already exist). The default is Ask.

:Disk Unit {*integer*} Disk onto which flod files will be copied. The default is 0.

:From Directory {*pathname*} Directory from which to copy files. On Symbolics 3600-family machines, the default is SYS:N-FEP;. On Ivory-based machines, the default is SYS:IFEP;.

:Hosts {*name*, All} Host(s) to which flod files will be copied. The default is your local FEP.

:Silent {Yes, No} Display files as they are copied. The default is Yes.

:Version FEP version. For example, G206, G208, V127, or I316.

Copy Microcode **Command**

Note: This command is implemented only on Symbolics 3600-family machines.

Copy Microcode *{version or pathname} destination keywords*

Installs a version of microcode.

version or pathname
Microcode version number or pathname to copy. *version* is a microcode version number (in decimal). *pathname* rarely needs to be supplied. It defaults to a file on FEP*n*:> (where *n* is unit number of the boot disk) whose name is based on the microcode name and version. (The file resides in the logical directory SYS:L-UCODE;.) The *version* actually stands for the file *appropriate-hardware*-MIC.MIC.*version* on FEP*n*:>.

destination FEP file specification. The pathname on your FEP*n*:> directory. The default is created from the microcode version.

keywords :Update Boot File

:Update Boot File
{*FEP-file-spec*, None, Query}. The pathname of the boot file you want it to update. The default is the current default boot file name.

Copy World **Command**

Copy World *file destination keywords*

Makes a copy of *file* (by default, a world load). This includes the specified world as well as any Incremental Disk Save (IDS) worlds on which it was built. See the section "Using the Incremental Disk Save (IDS) Facility". Copy World works from remote terminals. Copy World can also be used to copy netboot cores. You can boot a world from a remote world server with only a netboot core on your FEP. See the section "Netbooting".

file A FEP file specification; the world to copy. The default is constructed from the version of the world that you have booted.

destination A FEP file specification; the pathname for the new world. The default is a wildcard pathname assuring the correct hierarchical pathname relationship for the parent world and an IDS world.

Note: The `.ilod` file extension indicates world-load files for Ivory-based machines, just as the `.load` file extension indicates world-load files for Symbolics 3600-family machines. Files with the `.ilod` extension can be copied only between Ivory-based machines. Files with the `.load` extension can be copied only between Symbolics 3600-series machines.

After you issue the Copy World Command, Genera puts up a menu allowing you to specify the actions you want it to take:

```
There are 40450 blocks total in the files to be transferred.
  There are 809 blocks free on FEP1.
  You need 39641 more blocks.
  Possible actions to make space right now: Run dired  Expunge directory  Selectively delete

Parent IDS files to transfer: All parents  Missing parents  Just requested files  Selective
Update boot file to load FEP1:>Base-System-372-0.load.1: Yes  No
  Boot file to update: FEP0:>boot.boot.newest
  Update FEP0:>boot.boot.newest to load microcode 410: Yes  No
Transfer mode: Transfer-And-Checksum  Transfer-Only  Checksum-Only
Attempt automatic error recovery: Yes  No
<ABORT> aborts, <END> uses these values
```

Figure 106. Copy World

keywords :Automatic, :End Block, :File Set, :More Processing, :Output Destination, :Query, :Start Block, :Transfer Mode, :Update Boot File.

:Automatic {Yes, No} Whether or not to attempt automatic error recovery. The default is Yes.

:End Block {*integer*} The number of the last block to copy from source. The default is the last block, meaning copy until the end.

:File Set {All parents, Missing parents, Just Requested Files, Selective} Which parent IDS files to transfer. The default is Missing parents.

:More Processing {Default, Yes, No} Controls whether **More** processing at end of page is enabled during the output to interactive streams. The default is Default.

If No, output from this command is not subject to **More** processing.

If Default, output from this command is subject to the prevailing setting of **More** processing for the window. (See the section "FUNCTION M".)

If Yes, output from this command is subject to **More** processing unless it was disabled globally (see the section "FUNCTION M").

:Output Destination {Buffer, File, Kill Ring, Printer, Stream, Window} Where to redirect the typeout done by this command. The default is the stream ***standard-output***.

:Query {Yes, No} Whether or not to present a menu of transfer parameters. The default is Yes.

:Show Blocks Copied {Yes, No} Whether to print block numbers for blocks finished copying, every 100 blocks. The default is No, the mentioned default is Yes.

:Start Block {*integer*} The number of the block to start copying from the source. The default is 0, meaning begin at the beginning.

:Transfer Mode {Transfer-and-Checksum, Transfer-Only, Checksum-Only} Whether to verify the integrity of the copied world. The default is Transfer-and-Checksum. You can use Checksum-Only to checksum a band that you copied previously but were unable to checksum due to network problems.

:Update Boot file {*FEP-file-spec*, none}. Boot file to update to load the new world. The default boot file for IDS or complete worlds is boot.boot. The default for netboot cores is none.

Halt Machine **Command**

Halt Machine

Stops execution of Lisp and gives control to the FEP. You can now enter FEP commands, for example, to warm or cold boot the machine.

Show Machine Configuration **Command**

Show Machine Configuration *host keywords*

Shows the board-level hardware information about any Symbolics on the same network as your machine.

host The name of a Symbolics machine. The default is your machine.

keywords :More Processing, :Output Destination

:More Processing {Default, Yes, No} Controls whether **More** processing at end of page is enabled during the output to interactive streams. The default is Default.

If No, output from this command is not subject to **More** processing.

If Default, output from this command is subject to the prevailing setting of **More** processing for the window. (See the section "FUNCTION M".)

If Yes, output from this command is subject to **More** processing unless it was disabled globally (see the section "FUNCTION M").

:Output Destination
{Buffer, File, Kill Ring, Printer, Stream, Window} Where to redirect the typeout done by this command. The default is the stream ***standard-output***.

This information is useful for Symbolics Software Support Personnel. The display from Show Machine Configuration on a 3640 looks like this:

```
Chassis (PN 170219, Serial 70) in Chassis or NanoFEP:
    Manufactured on 10/6/84 as rev 1, functions as rev 1, ECO level 0
    Machine Serial Number: 4070
    FEP Version Number: 127
Datapath (PN 170032, Serial 2958):
    Manufactured on 1/7/85 as rev 3, functions as rev 3, ECO level 0
Extended Sequencer (PN 170299, Serial 720) in Sequencer:
    Manufactured on 5/12/86 as rev 1, functions as rev 1, ECO level 0
Memory Control (PN 170052, Serial 4102) in Memory Control or IFU:
    Manufactured on 6/11/86 as rev 5, functions as rev 5, ECO level 0
Floating Point (PN 170172, Serial 12) in FPA:
    Manufactured on 5/18/84 as rev 1, functions as rev 1, ECO level 0
Front End (PN 170062, Serial 3069) in FEP:
    Manufactured on 12/21/84 as rev 5, functions as rev 5, ECO level 0
512K Memory (PN 170002, Serial 1758) in LBus slot 00:
    Octal Base address: 0
    Manufactured on 9/28/84 as rev 2, functions as rev 2, ECO level 0
1Meg Memory (PN 170473, Serial 265) in LBus slot 02:
    Octal Base address: 4000000
    Manufactured on 10/3/85 as rev 1, functions as rev 1, ECO level 0
IO (PN 170157, Serial 849) in LBus slot 03:
    Octal Base address: 6000000
    Manufactured on 2/21/85 as rev 6, functions as rev 6, ECO level 0
512K Memory (PN 170002, Serial 1897) in LBus slot 04:
    Octal Base address: 10000000
    Manufactured on 10/15/84 as rev 2, functions as rev 2, ECO level 0
FEP Paddle Card (PN 170069, Serial 377) in FEP -- PADDLE side:
    Manufactured on 10/23/84 as rev 1, functions as rev 1, ECO level 0
IO Paddle Card (PN 170245, Serial 692) in LBus slot 03 -- PADDLE side:
    Manufactured on 6/17/85 as rev 1, functions as rev 1, ECO level 0
    Ethernet Address: 08-00-05-03-A0-15
```

FEP-Related Lisp Functions

These Lisp functions may be useful for system maintainers.

si:fix-fep-block *unit cylinder head sector* &key *:force-write-test* *Function*

Reads the block at the specified location reported to have an ECC error and optionally performs a read/write test which will likely correct the error. The function can be used to remove any bad blocks from use and place them in the bad-blocks file. The block numbers are printed in octal, so you should use #o when you type them.

If there is a read error, you are asked whether to try a write-read test. If rewriting results in no read error, the function exits, because the ECC error has been eliminated.

If a read error persists, or if you refuse the write-read test, you are queried about what to do with the block. There are four choices:

1. Splice it out.

2. Replace it with a block of zeros.

3. Replace it with a block containing a copy of the bad block.

4. Delete the file.

The first three choices result in the block being put in the bad-blocks file. This means that potentially good (no-media-defect) blocks can end up in the bad-blocks file. If there is an ECC error and you refuse the rewrite test, the block will be written to the bad-block file.

A similar function, **si:fix-fep-file**, tests all the blocks in a file.

Never use this function on a LMFS partition. For a similar function for use on LMFS partitions, see the function **lmfs:fix-file**.

si:fix-fep-file *pathname* *Function*

First reads all blocks in a file reported to have an ECC error and optionally performs a read/write test that will likely correct the error. The function can be used to remove these blocks from use and place them in the bad-blocks file.

Never use this function on a LMFS partition. For a similar function for use on LMFS partitions, see the function **lmfs:fix-file**. If there is a read error on a block, you are asked whether to try a write-read test. If rewriting results in no read error, the function exits, because the ECC error has been eliminated.

If a read error persists, or if you refuse the write-read test, you are queried as to what to do with the block. There are four choices:

1. Splice it out.

2. Replace it with a block of zeros.

3. Replace it with a block containing a copy of the bad block.

4. Delete the file.

The first three choices result in the bad block being put in the bad-blocks file.

Note that this means there is a way to put good (no-media-defect) blocks in the bad-blocks file. If there is an ECC error and the user refuses the rewrite test, thus not finding that the blocks have no defects, but chooses one of the first three options, the result is to write the blocks to the bad-block file.

A similar function, **si:fix-fep-block**, tests a single block.

si:machine-model *Function*

Returns at least two values: a keyword symbol designating the model number of a Symbolics computer, and a serial number for that computer. On a MacIvory, **si:machine-model** returns CPU major and minor revision numbers, as well.

Possible return values are as follows:

:unknown	The model number cannot be determined (usually indicating lack of some ID prom)
:/3600 or :|36xx**|**	(The keyword whose print-name is "36*xx*".) The machine is a Symbolics 3600 or 3600-family machine.
:XL400	The machine is a Symbolics XL400.
:|MacIvory|	The machine is a Symbolics MacIvory.
:UX400S	This is a Symbolics UX-family machine.

si:verify-fep-filesystem &optional (*unit* **0**) &key (*fix-checkwords* **':ask**) *Function*

Checks the FEP file system on disk unit *unit*, which defaults to zero, reporting any detected inconsistencies and offering to correct certain types of failures. If **:fix-checkwords** is **:ask** (the default), you are prompted if anything has to be fixed; the other options are **:yes** (always fix), **:no** (never fix), **:silently** (always fix without a message), and **:inform-only** (send messages only, do not fix, do not ask).

si:print-fep-filesystem &optional (*unit* **0**) *Function*

Outputs a textual description of the FEP file system on disk unit *unit*. The default value of *unit* is 0.

The **tape:write-fep-overlay-flods-to-cart** function may also be useful for site maintainers. For information about the **tape:write-fep-overlay-flods-to-cart** function, see the section "Using Lisp to Write Overlay (Flod) Files to Cartridge Tape".

Debugging in the FEP

You can enter "debug" mode in the FEP by issuing the Debug FEP command. This command is especially useful if you ever have problems that cause control to return to the FEP. (In the FEP, it's impossible to use the debugging methods normally used in Lisp.) See the section "Debug FEP Command".

Because the debug.flod file is not usually scanned in the hello.boot file, you may have to explicitly scan this flod file to make the Debug FEP command available. For more information, see the section "Scanning the Overlay (Flod) Files".

The Command Processor (CP) Copy Flod Files command will automatically copy the appropriate debug.flod file for your machine. Copy the debug.flod file *before* you need to use it; if you do not have debug.flod file on your FEP file system

(FEPFS), you will not be able to get it using Copy Flod Files once Lisp has crashed. For information about using the Command Processor (CP) Copy Flod Files command, see the section "Copy Flod Files Command".

Similar to when you're in the Lisp Debugger, you can move up and down the stack to attempt to determine the source of a crash. The HELP key lists available debug commands.

Debugging on 3600-family Machines

On 3600-family machines, if the machine has crashed during paging, use the c-m-S command to switch between the auxiliary stack (where paging code runs) and the normal stack (where user code runs). If the machine crashed while executing on the auxiliary stack, the c-m-S command will help you find user stack frames.

The Lisp function **si:show-machine-model** can be useful when debugging in the FEP. For more information about it,see the section "FEP-Related Lisp Functions". The Show Machine Configuration FEP command is also useful. For information about it,

see the section "FEP-Related Command Processor (CP) Commands".

Show Crash Data Command

Show Crash Data *keywords*

keywords :Interpreted, :Output Destination

:Interpreted Interprets the data in the current world.

:Output Destination
Directs the output of this command to a specified location. The location can be the pathname of a buffer or file, a printer, stream, or window.

Obtains the most recent output from the FEP's Show Status command, and makes it available in the Lisp world.

On 3600-family machines, this information includes some hardware state, the compiled code program counter (macro PC), virtual memory address (VMA), stack pointer and frame pointer (SP and FP), and the 16 most recent microcode program counters (OPCs).

On Ivory-based machines, the Mail Bug Report subcommand of the Debug FEP Command can be used to append the stack backtrace to the crash data obtained using the Command Processor (CP) Show Crash Data command.

On both 3600-family and Ivory-based machines, the information displayed by the Command Processor (CP) Show Crash Data command is similar to the information displayed by the Show Status FEP command. However, Show Crash Data provides an historical display; it shows the machine's status at the last crash. Show Status

reveals the machine's current status (that is, its status when the command was issued).

If you want to put all of the information from the Show Crash Data command in the current editor buffer or in a mail message, use the m-X Insert Crash Data Zmacs command. This makes it easy to include information about a Lisp crash that put you into the FEP. (This data is preserved even if you cold boot the machine, but not if you reset the FEP.)

The information retrieved from the Show Crash Data command is available until the next time Lisp halts, until the FEP is reset, or the machine is powered down. Therefore, even if you are unable to warm boot your machine, you can cold boot and then make a report with the data of the last crash.

Here is a possible sequence of events:

- The machine crashes and puts you into the FEP.
- The FEP issues the Show Status command automatically.
- On Ivory-based machines, you use the Debug FEP command and its Mail Bug Report subcommand.
- Unless this is a fatal error, you use the FEP command Start to warm boot the machine. If this does not work, use the Load World and Start FEP commands to cold boot the machine.
- You use the Command Processor (CP) Show Crash Data command in order to retrieve the most recent output of the Show Status FEP Command.
- You use the m-X Insert Crash Data command in the editor to save the information in a buffer.

Interpreting the Show Status Command's Output on 3600-Family Machines

The Show Status command displays the internal status of some Symbolics 3600-family machine registers. This display can provide information about machine states that cause the FEP message "Lisp stopped itself". In general, Show Status is not useful for interpreting wired-ferror halts.

When Show Status prints the contents of hardware registers, it decodes the bits symbolically. The FEP does not interpret these contents.

The Show Status output section "3600 program counters" includes a macro PC (the address of the current instruction of compiled Lisp code), a CPC (the address of the current microinstruction), and 16 OPCs (the addresses of the 16 most recently executed microinstructions; OPC+0 is the most recent, OPC+17 the earliest.

An arrow points to either the CPC or the first OPC, depending on the error condition that stopped the machine. This CPC or OPC pointed to represents the microinstruction that was executing at the time of the crash.

When interpreting the Show Status command's output to diagnose a crash, remember that:

- You must interpret some bits depending on the value of other bits.
- Some registers listed below are printed only if they contain "useful" information.
- The *Sequencer Status* and *MC Error Status* registers will be the most important ones to examine.

The "Show Crash Data Command" can automatically interpret some of this information for you, once the machine has been rebooted (and Lisp is running again). Use it with the :Interpreted keyword.

Self-Explanatory Hardware Errors

Bit	*Meaning*
Spare-error-bit	[never happens, unless manually wired to some signal]
GC-map-parity-error	GC MAP RAM on DP board.
Type-map-parity-error	TYPE MAP RAM on DP board.
Page-Tag-parity-error	PAGE TAG RAM on FEP board.
A-memory-parity-error	AMEM RAM on DP board.
B-memory-parity-error	BMEM RAM on DP board.
MC-error (map, ifu, or main mem)	Error on MC or IFU board; see MC/IFU error status.
AU-error	Error on AU (FPA) board (if the machine has one).
Task-state-memory-parity-error	TSKM RAM on SQ board (doesn't always halt machine).
Control-memory-parity-error	CMEM RAM on SQ board (also for microcode breakpoints if an L-Console program is cabled up for debugging).

Hardware "Errors" (not always really errors)

Bit	*Meaning*
CTOS-low-parity-error	CSTK RAM on SQ board (low half of output register).
CTOS-high-parity-error	CSTK RAM on SQ board (high half of output register).

Note: If CTOS-came-from-IFU is true, the above bits (CTOS-low-parity-error and CTOS-high-parity-error) have no meaning.

Sequencer Error Status

(Status of the SQ board and the main error status bits that can halt the machine)

The Microcode-halted bit will be set if a "halt" microinstruction was executed for one of the following reasons:

- A call to the %HALT function (due to a wired-ferror or a call to HALT).
- A fatal error, such as an error while entering the error handler or an error in wired code (page fault, disk handlers).
- Executing an undefined macroinstruction (running too old a version of microcode or executing bad macrocode).
- Failure of a microcode consistency check (stack frame too large, stack overwritten).

Sequencer Miscellaneous Status

(Status bits that are not errors)

Bit	*Meaning*
CTOS-came-from-IFU	CTOS register holds macroinstruction dispatch address from IFU (or TMC) rather than contents of CSTK RAM.
TSK-STOP (sequencer stopped)	Machine is stopped.
Errhalt-Sync	An error bit is set (stops machine).
MC Wait	Microinstruction waiting for memory control to allow the instruction to continue.
Task Switch	Switching to a different microtask.

ECC Syndrome

(An octal number followed by an address with x's in it)

This register contains the most recent main-memory read-error correction status. The error can be caused by a read by the processor, a read by the FEP, or a read by a DMA I/O device. The events that set this register include nonexistent memory reference, single-bit error correction, and double-bit error detection. Nonexistent memory and double-bit error halt the processor (even if it was the FEP or an I/O device that got the error). Currently, the FEP disables itself from getting a bus error if it references nonexistent Lbus memory or gets a double-bit error in Lbus memory.

Note: A bug in the FEP's code for examining the machine's status can set this register. In this case, the first two digits of the address are usually 77.

The address is the physical address of the location referenced. Only bits 23-18 and 1-0 are valid (the rest are x'ed out). These are sufficient bits to determine which

Lbus slot (bits 23-19) and which of the 8 banks within a memory board are being referenced. To convert the address to an Lbus slot number, consider the one or two digits at the left of the x's to be an octal value, and divide by 2. This is a logical slot number, as printed (in decimal) by the Show Configuration command. It is not related to the numbers printed on the machine chassis. Slot 0 is at the left, as seen from the front of the machine.

The syndrome codes are as follows:

```
000  okay   36      37      2-bit   38      2-bit   2-bit   3
010  39     2-bit   2-bit   6       2-bit   8       9       2-bit
020  40     2-bit   2-bit   13      2-bit   15      16      2-bit
030  2-bit  18      19      2-bit   20      2-bit   2-bit   unused
040  41     2-bit   2-bit   23      2-bit   25      26      2-bit
050  2-bit  28      29      2-bit   30      2-bit   2-bit   31
060  2-bit  33      34      2-bit   35      2-bit   2-bit   unused
070  NXM    2-bit   2-bit   unused  2-bit   unused  unused  2-bit
100  42     2-bit   2-bit   0       2-bit   1       2       2-bit
110  2-bit  4       5       2-bit   7       2-bit   2-bit   10
120  2-bit  11      12      2-bit   14      2-bit   2-bit   unused
130  17     2-bit   2-bit   unused  2-bit   unused  unused  2-bit
140  2-bit  21      22      2-bit   24      2-bit   2-bit   unused
150  27     2-bit   2-bit   unused  2-bit   unused  unused  2-bit
160  32     2-bit   2-bit   unused  2-bit   unused  unused  2-bit
170  2-bit  unused  unused  2-bit   unused  2-bit   2-bit   unused
```

3600 Program Counters

Label	*Meaning*
Macro PC	The address of the current compiled Lisp code instruction. This is prefaced with either (Odd) or (Even) since there are two instructions per word.
Current micro PC (CPC)	The address of the current microinstruction.
Old PCs (OPC)	The addresses of the 16 most recently executed microinstructions. OPC 0 was executed most recently, OPC 17 least recently.

FEP buffer status

Bit	*Meaning*
Spy DMA Enb	Spy bus being used by FEP to access disk or net (means spy bus being used for normal functions).
Write to / Read from Device	Spy DMA direction.
Drive Busy	Spy DMA Mode (who controls busy line).
Int Enb	Spy DMA Enable to interrupt FEP.
Count Up / Count Down	Spy DMA address increment direction.
Busy	Spy DMA Busy (inside FEP).
Spy DMA Busy	Spy DMA busy line (on backplane).
DMA Setup	[meaning unknown].

FEP Lbus control

Bit	*Meaning*
ECC Diag	Normal memory error correction logic disabled; instead, FEP can read or write the 8 extra bits of main memory.
Doorbell Int Enb	Doorbell (Lisp-to-FEP signal) interrupt enabled.
Use Uncorrected Data	FEP unaware of corrected Lbus data if single-bit-error.
Ignore Double ECC Error	FEP does not get bus error if uncorrectable Lbus error (either double-bit error or nonexistent memory).
Task 3 Req	FEP trying to wake up microtask 3.
Doorbell	Doorbell ringing (Lisp-to-FEP signal).
Lbus Buffer Busy	[self-explanatory].
Lbus Buffer Some Parity Error	[self-explanatory].

FEP Board ID control

Bit	*Meaning*
Continuity	Read-back of random signal that checks board presence.
Lbus ID Req	Lbus reading board IDs during not-normal functions.
Half Speed	Main processor clock running at half speed.

FEP Process Control

Bit	*Meaning*
Lbus Power Reset	Reset all Lbus devices due to power on or power off.
Lbus Power Reset (on bus)	Same as above, but actually read back from the bus.
Lbus Reset	Reset all Lbus devices.
Lbus Reset (on bus)	Read back the Lbus Reset.
Clear Errors	Bit that clears FEP error registers (not an error).
FEP Int Enable	FEP interrupt enable (not an error).
Kept Alive	FEP died and was reset by nanofep.
FEP RAM Par Err	Parity error in dynamic RAM on FEP board.

MC / IFU Error Status

Bit	*Meaning*
Double bit error	An uncorrectable error in main memory, or a reference to a nonexistent Lbus address. See "ECC syndrome".
Map A parity error, Map B parity error	Parity errors in the map caches on the MC (TMC, IFU) board.
Hit in both Map A and Map B	Both map caches claim to map the same address. Could be map hardware or microcode problem causing bad data write
IFU Op Parity Error	Parity error in internal IFU operation.
IFU Arg Parity Error	Parity error in internal IFU argument.

Decoding Micro PCs on 3600-Family Machines

On Symbolics 3600-family machines, you can use the Lisp functions **dbg:decode-micro-pc** and **dbg:decode-micro-pcs** to decode the microcode PCs printed by the Show Status FEP command. You can use the "Show Crash Data Command" to automatically decode micro PCs in crash data (use the :Interpreted keyword). For more information about crash data, See the section "Show Crash Data Command".

dbg:decode-micro-pc *pc* &optional (*name* **sys:%microcode-version**) (*version* **(sys:microcode-version-number sys:%microcode-version)**) *Function*

Useful for investigating why a machine crashed. It decodes an octal microinstruction address printed by the Show Status FEP command.

To use this function, write down the Show Status FEP command's output. Then, either warm boot the machine using the Start command, or call **dbg:decode-micro-pc** on another machine. To decode more than one octal microinstruction address, see the function **dbg:decode-micro-pcs**.

dbg:decode-micro-pcs *pcs* &optional *(name* **sys:%microcode-version***) (version* **(sys:microcode-version-number sys:%microcode-version)***) verbose (load-symbol-table* **:ask***)* *Function*

Useful for investigating why a machine crashed. It decodes the octal microinstruction addresses printed by the Show Status FEP command.

To use this function, write down the Show Status FEP command's output. Then, either warm boot the machine using the Start command, or call **dbg:decode-micro-pcs** on another machine. To decode only one octal microinstruction address, see the function **dbg:decode-micro-pc**.

pc is an address in the microcode, taken from the CPC or OPC information printed (in octal) by the Show Status FEP command. If your default radix is decimal, precede *pc* by **#o**.

Normally the number in the Show Status FEP command's output with the arrow (→) pointing to it is the relevant one, but you may want to decode all of the numbers for additional clues.

name and *version* are optional; they specify the version of the microcode that was running at the time of the crash. Omit these arguments if you call **dbg:decode-micro-pc** or **dbg:decode-micro-pcs** while using the machine that crashed (make sure that you're running the same microcode version that was running at the time of the crash).

Also, omit these arguments if you call the functions from another machine that has the identical software *and* hardware configuration to that of the machine that crashed. To find a machine's microcode version name and number, use the Command Processor (CP) Print Herald command with the keyword :Detailed, or look for the name and version number of the microcode file from the machine's boot file (normally, this is file fep0:>boot.boot).

Microcode version numbers are decimal; include a period at the end of the number if your default radix is octal.

dbg:decode-micro-pc and **dbg:decode-micro-pcs** print information that depends on the microinstruction:

Microinstruction	*Information printed*
Halt instruction	The reason it halted the machine. An example is "error in the error handler". Reasons are provided as constant strings in the microcode source program. They do not represent any dynamic analysis of the machine's state.
Signaller of a Lisp error	The internal form of the error message. Normally, Lisp software translates error messages into conditions and signals them. The conditions define more readable error messages. This is useful mainly in decoding OPCs other than ones with the arrow provided by the Show Status FEP command.

Handler for a macroinstruction in compiled Lisp code

The name of a macroinstruction. Might be caused by running a world with an incompatible microcode (such as a microcode from an earlier release).

If all else fails, the functions offer to load the microcode symbol table (from the SYS:L-UCODE; directory) and then print the symbolic name of the microinstruction. It takes a few minutes to load the table. Microinstruction symbolic names can sometimes contain clues about what the machine was doing when it crashed.

When the symbolic name of the microinstruction includes parentheses, it is a list containing the name of a microcode routine and the path through that routine by which to reach the microinstruction in question. Remember that these names are not unique; the same microinstruction may be reachable via multiple paths, and from different microcode routines. For example, a microinstruction named (FTN-AR-1 3) might also be part of the microcode for the CAR instruction.

When the name has no parentheses, it is unique; it names the first microinstruction of a microcode routine.

Note: If the reason Lisp stopped itself is anything other than "microcode halted", the information that the Lisp functions **dbg:decode-micro-pc** and **dbg:decode-micro-pcs** print will be most useful for people who understand Symbolics hardware.

Decoding Macro PCs on 3600-Family Machines

To decode the macrocode PC printed by the Show Status FEP command, warm boot, if you can. Then, use the "Show Crash Data Command" to automatically decode macro PCs in crash data (use the :Interpreted keyword). For more information about crash data, see the section "Show Crash Data Command".

If you cannot warm boot, go to another 3600-family machine running identical software, and call the function **sys:%find-structure-header** on the number printed by the FEP. This is an octal number; use **#o** if necessary. It returns a compiled-function object, which is the function that was executing at the time.

To find the exact place in the function that was executing, note the difference between the number printed by the FEP and the address in the printed representation of the compiled-function object. You can use **sys:%pointer-difference** to compute this difference.

Multiply this by 2, and add 1 if the PC was odd (not even). The result is the instruction number of the current instruction; disassemble the compiled function to see it.

Example:

```
FEP Command:  Show Status
...
3600 program counters:
  Macro PC/ (Odd)1244531
  ...
FEP Command: Start
...
(%find-structure-header #o1244531)
#<DTP-COMPILED-FUNCTION EQUAL 1244530>
(%pointer-difference #o1244531 *)
1
(1+ (* * 2))
3
(disassemble ***)
  0 ENTRY: 2 REQUIRED, 0 OPTIONAL
  1 PUSH-LOCAL FP|0             ;A
  2 PUSH-LOCAL FP|1             ;B
  3 BUILTIN EQL STACK
  ...
```

Instruction 3 (EQL) is the one that halted.

Debugging on Ivory-based Machines

This section describes how to use the Ivory FEP (IFEP) Debugger. For information on debugging 3600 machines, see the section "Debugging on 3600-family Machines".

The main use of IFEP Debugger is to file bug reports about system errors that are not handled by the Lisp Debugger (for example, errors in the storage system, or in wired interrupt handlers).

The IFEP Debugger is used internally by Symbolics to diagnose low-level system errors, or errors that occur before the Lisp Debugger is loaded.

Because the IFEP Debugger has access to Lisp virtual memory, you can also use the IFEP Debugger to modify Lisp variables or other structures.

Note that to use the IFEP Debugger to diagnose system problems or to set Lisp variables, you must be familiar with the machine architecture and the internal representation of Lisp objects.

When Lisp stops because of a error, it invokes the FEP. The FEP prints the error message supplied by Lisp (or prints the machine state, if Lisp stopped because of an undetected error), and then suggests that you use the IFEP Debugger to record backtrace information for a bug report.

```
Lisp Stopped Itself
<Lisp error message, if any>
Use :Debug and c-M to record a backtrace.
FEP Command:
```

Using the IFEP Debugger to Report System Errors

The IFEP Debugger is useful when Lisp stops itself because of an error, or the machine reboots (to the FEP). You can also use the Debugger when the machine hangs and does not respond to keyboard interrupts.

If Lisp stops because of a system error, the FEP prints the error message provided by Lisp, and suggests that you use the :Debug command. If the system hangs and does not respond to the normal keyboard interrupts, you can use h-c-FUNCTION to stop Lisp. (You may need to press h-c-FUNCTION more than once, if Lisp is running uninterruptably.) As a last resort, you can reset the machine to reboot to the FEP. A fatal error will cause Lisp to reboot to the FEP.

The standard procedure for reporting system errors using the IFEP Debugger is as follows:

1. Invoke the IFEP Debugger using the :Debug command. See the section "Debug FEP Command".

2. Use other IFEP Debugger commands and command accelerators to isolate and display relevant backtrace data. See the section "IFEP Debugger Command Descriptions".

3. Record a backtrace in the "crash data" area, using the :Mail Bug Report (c-M) command. You can alternatively use the :Output Destination keyword option to record crash data in a file. You can include this data in a Lisp system bug report.

4. Leave the IFEP Debugger and resume Lisp execution using :Abort (c-Z or ABORT).

5. Compose and send the bug report using the :Report Bug CP command or the (m-X) Report Bug Zwei command.

   If you recorded your crash data in an :Output Destination file, you should include this file when you compose your bug report. If you recorded the backtrace in the "crash data" area, use the m-X Insert Crash Data command when you compose your bug report.

Entering the IFEP Debugger

Invoke the IFEP Debugger by entering the following command at the FEP command prompt.

```
:Debug
```

You can use *keyword* options to specify IFEP Debugger initial conditions (for example, :Ignore Storage Structures Yes, or :Show Initial Frames No). See the section "Debug FEP Command".

If the IFEP Debugger (or the FEP itself) appears confused (for example, if the display appears erratic or if there is a lot of error printing), the system error may have corrupted the FEP's state information image or data. In this case, first reboot the FEP before you proceed. Do *not* boot Lisp directly; this could destroy the state information necessary for debugging the error. For additional information on resetting and booting the FEP (without booting Lisp), see the document *Genera 8.1 Software Installation Guide*. If the machine hangs and does not respond to any of the usual keyboard interrupts (including h-c-FUNCTION), or if the machine reboots (to the FEP), you might still be able to do some debugging. Since Lisp's state is preserved by Boot ROM on reset, you can still debug, even if the machine reboots or if you have to reboot by resetting the machine (hardware).

Sending a Bug Report Using the IFEP Debugger

Use the IFEP Debugger command :Mail Bug Report (c-M) to record a backtrace of the error. Keyword options allow you to limit the size or detail of the backtrace, or to save the backtrace to a disk file. See the section "Mail Bug Report IFEP Debugger Command".

Note that this command only captures backtrace information. It does *not* mail a Lisp bug report. When you are next running Lisp, you can use the Report Bug command to compose the actual bug report. See the section "Report Bug Command".

To include an IFEP Debugger backtrace in your Lisp bug report, enter (m-X) Insert Crash Data, or (m-X) Insert File when you are in the Zwei bug-report window. (If you have saved the backtrace to a file, the "crash data" will be the name of the file containing the backtrace.)

The following example demonstrates how to include an IFEP Debugger backtrace in a Lisp bug report:

```
FEP Command:  :Debug
→  c-M  :Mail Bug Report Nframes 3 :Output Destination crash-data.text
→  :Abort
FEP Command:  :Start
Command:  Report Bug
m-X Insert File crash-data.text
```

Note that you must enter the Lisp Report Bug command to actually report the bug. You can insert the file containing the crash data with the (m-X) Insert File command in the minibuffer.

Exiting the IFEP Debugger

To exit from the IFEP Debugger (and return to the top-level FEP command processor) type the following IFEP command at the prompt:

```
→ :Abort
```

You can also press one of its two command accelerators, c-Z or ABORT. Note that ABORT returns you to the FEP only when you enter it at the top-level IFEP Debugger prompt. Otherwise, ABORT terminates the current activity (for example, the execution of a command, or the typing of command input).

If the error that caused Lisp to halt is a proceedable error, the IFEP Debugger reminds you that you can resume execution of Lisp using the Continue FEP command, after you exit the IFEP Debugger (see the section "Continue FEP Command").

A proceedable error is a non-fatal error that is detected when the normal error reporting system is not usable. In this case, Lisp halts to the FEP so that the error can be recorded. Continuing will clear the error in a way that does not damage Lisp.

Often, you can clear *non-proceedable* Lisp errors by warm booting, using the Start FEP command (see the section "Start FEP Command"). (Note that you may need to use :Ignore Saved State Yes to abort any unwind-protects or binding restoration in the erring process.)

If all else fails, you can use the Boot FEP command to cold-boot Lisp (see the section "Boot FEP Command"). The FEP will preserve your "crash data" backtrace (until you create another backtrace) as long as the FEP is not reset (either manually or because of a fatal error).

Even if you crash again before you mail the bug report, the original backtrace will still be available. If you have already reported the bug which included the backtrace, you can reset the FEP to recover the memory used by the backtrace. For more information, see the section "Reset FEP FEP Command".

Using IFEP Debugger Commands

The IFEP Debugger offers more than thirty full-form commands and command accelerators. You can use these commands to study the Lisp world and to record stack information for further analysis.

In general, IFEP Debugger command functions and syntax are similar to Lisp Debugger commands. Differences between similar IFEP and Lisp Debugger commands are noted in the description of each individual command (see the section "IFEP Debugger Command Descriptions").

The IFEP Debugger does *not* have an evaluator. Instead it supports additional commands that perform debugging functions normally provided in Lisp. The IFEP Debugger also supports DDT-like commands, which allow you to examine Lisp memory locations for symbol values, functions, structure slots, and instance variables or slots.

IFEP Debugger commands use a concept of *point* (also called *dot*), which means the location (address and segment) being examined. Most of the IFEP Debugger commands and accelerators use point in some way. Many commands use point as their default argument. Some of the IFEP Debugger commands set point. For more details, see the section "IFEP Debugger Command Descriptions".

When you are using IFEP Debugger DDT commands, a location being examined is considered *open* (meaning that you could change its contents). The IFEP Debugger indicates that a location is open by leaving the cursor on the current line, immediately after displaying the contents of the location. That is, the IFEP Debugger does not display a new prompt. Pressing RETURN closes the open location and returns you to the Debugger prompt.

You can use other IFEP Debugger DDT commands to operate on the current open location (for example, to set a Lisp variable), or to close the current location and open another.

Getting Help in the IFEP Debugger

The IFEP Debugger assumes familiarity with system architecture and the internal representation of Lisp objects. Consequently, IFEP Debugger self-documentation is minimal.

At the command prompt level, you can use HELP to display a brief description of IFEP Debugger commands, accelerators, or modifiers. You can also use c-/ or c-? to see a list of the commands matching what you have typed so far.

Since the accelerators echo their full-form command, you can use the full form to get help. Note that some IFEP Debugger commands (the Point Stack accelerators), do not have full-form command equivalents (for more details, see the section "IFEP Debugger Point Stack Accelerators").

c-HELP or :Accelerator Help gives a brief paragraph describing accelerator arguments and modifiers, and then lists the accelerator keys and their names. (Usually the accelerator key name is similar to the full-form command that the accelerator invokes.)

Aborting IFEP Debugger Commands or Output

You can usually stop a IFEP Debugger command by pressing c-ABORT.

If the Debugger appears to be looping and c-ABORT does not respond, you can use h-c-FUNCTION to force the Debugger back to its command-level prompt.

If you want to stop the IFEP Debugger while it is producing output (for example, printing), press any key. This aborts the output and returns to the top-level prompt. Note that, if you press ahead, you might accidentally abort the output of a command. If the command output appears truncated, you can usually recover by reentering the command.

Entering IFEP Debugger Commands

Like the Lisp Debugger, the IFEP Debugger prompt is →. The IFEP Debugger accepts both full-form commands (such as :Show Backtrace), and command accelerators (such as c-B).

Full-form IFEP Debugger commands begin with a colon (:) or m-X. When you press either of these, the Debugger indicates that it expects a full-form command by displaying the following prompt

```
Debugger command:
```

Full-form commands use positional and keyword arguments and command completion, as in the rest of the FEP (and Genera). Accelerators take their arguments in a fashion similar to the Lisp Debugger (or Zmacs editor).

Note that the IFEP Debugger indicates that a location is open by not prompting for a new command. See the section "Using IFEP Debugger Commands".

Using IFEP Debugger Command Accelerators

Most IFEP Debugger commands have associated command accelerators. For those commands you can invoke an IFEP Debugger command either by typing the full-form command or by entering the associated command accelerator. See the section "IFEP Debugger Command Descriptions".

The IFEP Debugger Point Stack commands are invoked only through accelerators. That is, they do not have full command equivalents. For more details, see the section "IFEP Debugger Point Stack Accelerators".

IFEP Debugger accelerator commands may take command modifiers and arguments and argument modifiers. Accelerator argument modifiers further specify accelerator arguments. Accelerator command modifiers are interpreted differently for individual IFEP Debugger commands (some commands can take an additional argument when prefixed by this argument).

IFEP Debugger Accelerator Arguments

The IFEP Debugger interprets anything that is not a full-form command or a command accelerator as an accelerator argument.

The IFEP Debugger supports the following types of accelerator arguments:

- Numeric
- Symbolic
- Typein

IFEP Debugger accelerator arguments can be further specified using accelerator argument modifiers. See the section "IFEP Debugger Accelerator Argument Modifiers".

Numeric Arguments

Numeric arguments are similar to Zwei numeric arguments. Any of the digit characters (in the current base), +, -, c-U, or ∞ indicate an IFEP Debugger numeric argument. If you enter a numeric argument with any of c-, m-, s-, or h- keys held down, the numeric argument is parsed in base 10. Numbers are parsed in the current number base (which defaults to 8). You can change the number base using the "Set Base IFEP Debugger Command".

The character "." adds the value of point (or dot) to any accumulated numeric argument.

Note that the argument parser does not do arithmetic. A minus sign (-) simply negates the accumulated argument. For example, the following numeric arguments are equivalent:

```
.+3, 3+., +.3, 3.+, 3.
```

These arguments are also equivalent,

```
.-4, 4-., -.4, 4.-
```

Also note that the following argument is -34, not -1.

```
c-3 c-- c-4
```

Symbolic Arguments

Characters that cannot be parsed as numeric are parsed as symbolic arguments. In this case, the Debugger reminds you that you are entering a symbolic name by reprompting with:

```
Examine (symbol)
```

Symbolic names include all the interned symbols in Lisp, as well as those symbolic names explicitly defined by the IFEP Debugger.

Currently, the only symbolic names defined by the IFEP Debugger are the names of the machine registers. They are of the form:

```
%REGISTER-XXX
```

Register symbolic name accelerator arguments are translated to the appropriate address in the register segment (as specified in the IFEP debugger command argument modifier or in the :Set Default Segment command). See the section "IFEP Debugger Accelerator Argument Modifiers" and see the section "Set Default Segment IFEP Debugger Command".

If an IFEP Debugger symbolic name collides with a Lisp symbol, you can enter the Lisp symbol by explicitly specifying the Lisp symbol package.

Note that when you specify a package you must separate it from the symbol with a colon (:). See the section "Set Package IFEP Debugger Command".

Symbol name prefixes specify how to interpret the symbol name.

#' Uses the symbol function-cell contents.
' Uses the address of the symbol.

If you do not enter a prefix, the accelerator argument is interpreted as the symbol variable value.

Typein Arguments

A typein argument allows you to type an argument in a format similar to one of the typeout formats (see the section "Using IFEP Debugger Typeout Formats").

Typein arguments begin with a backquote (‘) in IFEP Debugger commands. The IFEP Debugger supports the following typein formats:

‘Q An argument created from a cdr-code, data type, and pointer

‘O A number (in the current base)

‘S A symbol

Note that the ‘Q typein mode prompts for a cdr-code, data-type (both symbolic), and a pointer. The IFEP Debugger then constructs the appropriate accelerator argument.

‘O and ‘S typein modes explicitly instruct the Debugger to parse a numeric or symbolic argument, respectively.

IFEP Debugger Accelerator Argument Modifiers

IFEP Debugger argument modifiers begin with an atsign (@).

Most IFEP Debugger command accelerators treat their arguments as an address or location. IFEP Debugger accelerator argument modifiers for these arguments allow you to specify one of the following segments for that address to be interpreted in.

@V Virtual memory

@P Physical memory

@R Register number

@U "Unmapped" memory

The @U modifier identifies a subset of virtual addresses (that is, those virtual addresses that map directly to physical addresses.) These are also called **vma-equals-pma**. Since the wired system operates in **vma-equals-pma**, it is often useful to be able to specify a **vma-equals-pma** in terms of the physical address that it would map to.

Note that when you use the @R modifier, the addresses in the R segment do not correspond to the internal register numbers of the Ivory chip. They correspond to virtual register numbers. Only the registers that are available in the FEP Show Status command are accessible.

The best way to access registers is to use the symbolic register name. See the section "Set Default Segment IFEP Debugger Command". The self-documentation for this command's argument gives the possible segments and their abbreviations.

IFEP Debugger Accelerator Modifiers

The lozenge (◊, sh-ESCAPE, sy-ESCAPE, or ESCAPE) identifies an IFEP Debugger accelerator command modifier. Accelerator commands interpret this prefix differently. In general, you can use this modifier to specify a second accelerator argument. For example, the following command specifies two numeric arguments:

Here are some examples of using ESCAPE to give a second argument to an accelerator or modify its defaults. Notice that each accelerator interprets ESCAPE differently:

```
→ c-m-D :Show Compiled Code -133936248 :Radius 4
Disassembled code for #<DTP-COMPILED-FUNCTION SI:AUX-HALT 37001045610>:
=>   0  ENTRY: 0 REQUIRED, 1 OPTIONAL
     2  PUSH NIL
     4  START-CALL-INDIRECT-PREFETCH #'SI:AUX-WAIT-FOR-DISK-DONE
```

The default arguments to :Show Compiled Code (c-m-D) are the current function and a radius of 4. (For more details, see the section "Show Compiled Code IFEP Debugger Command".)

To see more, use ESCAPE c-8 to keep default first argument, and change second argument to 8:

```
→ESCAPE c-8 c-m-D :Show Compiled Code -133936248 :Radius 8
Disassembled code for #<DTP-COMPILED-FUNCTION SI:AUX-HALT 37001045610>:
=>   0  ENTRY: 0 REQUIRED, 1 OPTIONAL
     2  PUSH NIL
     4  START-CALL-INDIRECT-PREFETCH #'SI:AUX-WAIT-FOR-DISK-DONE
     6  FINISH-CALL-0-EFFECT
    10  PUSH-CONSTANT '#<DTP-LOCATIVE to SYS:*LISP-STOPPED-CLEANLY*
37001010062>
     7  PUSH T
→
```

Similarly, you can give an Nframes argument of 5 to (c-M):

```
→c-5 c-M :Mail Bug Report :Nframes 5 :Output-destination NIL
```

Adding an ESCAPE here changes the output destination to a file:

```
→ c-5 ESCAPE c-M :Mail Bug Report :Nframes  5 :Output-destination
#P"FEP0:>Crash-data.text.newest"
```

See the section "IFEP Debugger Command Descriptions" for details on how to use this modifier in particular IFEP Debugger commands.

IFEP Debugger Command Descriptions

IFEP Debugger commands are functionally classified as follows:

- Stack Display Commands
- Stack Motion Commands
- Point Stack Accelerators
- DDT Commands
- Lisp-like Commands
- Miscellaneous Commands

Command Summary

This section briefly describes IFEP Debugger commands and keywords. Default keyword options are shown in **bold**.

Stack Display Commands

These commands display backtrace information and information about the current stack frame (such as argument values, local variable values, disassembled code, and so on).

Accelerator	*Command*	*Keywords or Arguments*
c-m-A	:Show Argument	Argument {base 10 integer}
c-B, m-B	:Show Backtrace :N Frames {base 10 integer} :Step {Yes, **No**}	:Detailed {Yes, **No**}
c-m-D, c-X D c-X c-D	:Show Compiled Code	:PC {a relative PC} :Radius {Base 10 integer}
c-L, m-L c-X c-L c-X c-A	:Show Frame	:Clear Window {Yes, **No**} :Detailed {Yes, **No**}
c-m-F	:Show Function	
c-m-L	:Show Local	Local {base 10 integer}

Stack Motion Commands

These are stack navigation commands.

Accelerator	*Command*	*Keywords or Arguments*
m->	:Bottom of Stack	:Detailed {Yes, **No**}
c-S	:Find Frame	Frame {string} :Detailed {Yes, **No**} :Reverse {Yes, **No**}
c-N, m-N	:Next Frame :N Frames {base 10 integer}	:Detailed {Yes, **No**}
c-P, m-P	:Previous Frame :N Frames {base 10 integer}	:Detailed {Yes, **No**}
m-<	:Top of Stack	:Detailed {Yes, **No**}

Point Stack Accelerators

These accelerators maintain the Point Stack (or PDL). Note that they have have no IFEP Debugger command equivalent. They correspond closely to the point-pdl commands in Zmacs (c-SPACE for (m-X) Set Pop Mark, and c-m-SPACE for (m-X) Move to Previous Point).

Accelerator	*Description*
0 c-SPACE	Displays the Point Stack
c-m-SPACE	Exchanges point and top of stack
c-U c-U c-SPACE	Pops a location off the stack and discards it
c-U c-SPACE	Pops a location off the stack and sets point
c-SPACE	Pushes point pdl onto the stack
n c-m-SPACE	Rotates the top *n* entries of the point stack

DDT Commands

These commands perform basic Debugger functions (similar to conventional DDTs).

Accelerator	*Command*	*Keywords or Arguments*
=	:Describe Location	Address {an address}
	:Print Location {Yes, **No**}	
	:Segment {a segment}	
^, RETURN	:Set Location Contents	Address {an address}
LINE, SPACE		Tag {an integer}
BACK-SPACE		:Follow Forwarding {Yes, **No**}
	:Format {a typeout format}	
	:Segment {a segment}	
	:Then Show {an address or Null}	
	:Set Typeout Format	Format {a typeout format}
/, ^, TAB	:Show Location Contents	Address {an address}
LINE, SPACE		:Follow Forwarding {Yes, **No**}
BACK-SPACE		:Format {a typeout format}
	:Print Location {**Yes**, No}	
	:Segment {a segment}	
;	:Show Value in Format	Tag {an integer}
	Pointer {an address}	
	:Format {a typeout format}	

Use these typeout formats with the :Format keyword in the IFEP Debugger DDT commands to specify a display format for a value.

'	Character
#	Bit Number
A	Array Header
C	Control Register
E	Error Trap
I	Instruction
O	"Octal"
Q	Lisp Pointer
S	Lisp Object

Lisp-like Commands

These commands perform debugging functions normally provided in Lisp.

Accelerator	*Command*	*Keywords or Arguments*
	:Describe Area	Area {an area name}

:Describe Physical Address	Address {an address}
:Describe Region	Region {a region number}
:Describe Virtual Address	Address {an address}
:Symbol Function	Symbol
:Symbol Value	Symbol {interned Lisp symbol}

Miscellaneous Commands

These commands do not fit in the other IFEP Debugger command functional categories.

Accelerator	*Command*	*Keywords or Arguments*
c-Z	:Abort	
c-HELP	:Accelerator Help	
	:Debug Process	Process {name and address}
	:Show Initial Frame {**Yes**, No}	
c-M	:Mail Bug Report	Nframes {base 10 integer}
	:Output Destination	{FEP pathname, or Null}
	:Set Base Number Base {base 10 integer}	
	:Set Debugger Options	:Follow Forwarding {Yes, **No}**
	:Name Heuristication {**Yes**, No}	
	:Print Errors {Yes, **No**}	
	:Print Length {an integer}	
	:Print Level {an integer}	
	:Set Default Segment	Segment {a segment}
	:Set Package New Package {a package}	

Stack Display Commands

These commands display backtrace information and information about the current stack frame (such as values, local variable values, disassembled code, and so on).

Show Argument IFEP Debugger Command

:Show Argument *Argument* c-m-A

Shows an argument in the current frame. This command sets point to the stack-address of the argument.

For example,

```
STORAGE:CHECK-UNIT-HUNG (PC = 70)
    Arg 0 (UNIT): #<STORAGE:EMBEDDED-UNIT-QUEUE 36000603042>
→c-m-0 c-m-A  :Show Argument 0
#<STORAGE:EMBEDDED-UNIT-QUEUE 36000603042>
→TAB
36000603042@V/04170200000060  ;s  #<DTP-HEADER-I 30200000060>
→
```

Since c-m-A sets point to the argument, you can use TAB> to examine the structure (in this case) further.

Argument {*base 10 integer*} Argument number (from :Show Frame display). (Default is 0.)

Key-binding accelerators

Use c-L or c-X c-A to show all the arguments in the frame.

c-m-A :Show Argument 0

Show Backtrace IFEP Debugger Command

:Show Backtrace *keywords* c-B, m-B, c-m-B

Displays a backtrace. This command does not change the value of point.

Note that the IFEP Debugger does not have a concept of invisible or interpreter frames. They are always shown, as are continuations.

keywords :Detailed, :N Frames, :Step

:Detailed {Yes, No} Shows the arguments and current PC for each frame. The default is No (Yes, if mentioned).

:N Frames {*base 10 integer*} Specifies how many frames to include in the backtrace. (Default is 5.)

:Step {Yes, No} Stops after each frame.

In :Detailed Yes mode, :Step Yes prompts for "Next Frame?", to which you can reply Yes or No.

In :Detailed No mode, there is no prompt. Pressing RUBOUT, ABORT, or N will stop. Any other character continues. (Default is No. Mentioned default is Yes.)

Key-binding accelerators

c-B	:Show Backtrace
m-B, c-m-B	:Show Backtrace :Detailed Yes :Step Yes

Show Compiled Code IFEP Debugger Command

:Show Compiled Code *Function keywords* c-m-D, c-X D, c-X c-D

Shows the compiled code (that is, disassembled instructions) around an address.

Note that this command takes different arguments than the similar Lisp Debugger command.

This command sets point to the relative PC in the specified function.

Function	{*an address in a compiled function*} Note you can enter a PC as an argument to c-m-D. The default is the current PC of the current frame.
keywords	:PC, :Radius
:PC	{*a relative PC*} The PC to center the disassembly on. (Default is the PC pointed to by *function*.)
:Radius	{*base 10 integer*} The number of words to display before and after the PC. (Default is 4.)

Key-binding accelerators

Note that the IFEP Debugger accelerators c-x D and c-m-D are compatible with the Lisp Debugger (c-X c-D is :Show Source Code in the Lisp Debugger).

You can use the accelerator modifier to specify a different radius. For example, ESCAPE c-8 c-m-D is equivalent to :Show Compiled Code <default> :Radius 8.

c-X c-D :Show Compiled Code <accelerator argument> :Radius 4

c-m-D

c-X D

Show Frame IFEP Debugger Command

:Show Frame *keywords* c-L, m-L, c-X c-L, c-X c-A

Shows the current stack frame function and arguments. Sets point to the first stack-address (FP|0) of the frame.

keywords	:Clear Window, :Detailed
:Clear Window	{Yes, No} Clears the window first. (Default is No. Mentioned default is Yes.)
:Detailed	{Yes, No} Shows locals and disassembled code, as well as any bindings in the frame. You can use this command option to discover the value of specials in other processes. This is equivalent to the Lisp Debugger command :Show Stack. (Default is No. Mentioned default is Yes.)

Key-binding accelerators

c-L, c-X c-A	:Show Frame :Clear Window Yes
m-L, c-X c-L	:Show Frame :Detailed Yes :Clear Window Yes

Show Function IFEP Debugger Command

:Show Function c-m-F

Shows the function in the current frame.

Sets point to the function-cell of the function.

Key-binding accelerators

c-m-F	:Show Function

Show Local IFEP Debugger Command

:Show Local *Local* c-m-L

Shows a local variable in the current frame. You can also use this command to show any values accumulating in the frame. (This is equivalent to :Show Frame :Detailed.)

Note that you can use this command to examine any slot in the current frame (the IFEP Debugger does not distinguish between a local variable and an internal stack slot). You can display an *&rest* argument using :Show Local with an index one less than the first local.

Sets point to the stack-address of the local.

Local {*base 10 integer*} Local number (from :Show Frame :Detailed display). (Default is 0.)

Key-binding accelerators

Use m-L or c-X c-L to show all the locals in the frame.

c-m-L :Show Local <accelerator argument>

Stack Motion Commands

These commands move to another activation frame in the stack (note that as in the Lisp Debugger, *down* (:Next Frame) is less recent, or toward the caller, and *up* (:Previous Frame) is more recent, or toward the callee):

Bottom of Stack IFEP Debugger Command

:Bottom of Stack *keywords* m->

Goes to the bottom of the stack (that is, the least recent frame). Sets point to the first stack-address (FP|0) of the frame.

keyword :Detailed

:Detailed {Yes, No} Shows locals and disassembled code for the frame. (Default is No. Mentioned default is Yes.)

Key-binding accelerators

m-> :Bottom of Stack

Find Frame IFEP Debugger Command

:Find Frame *Frame keywords* c-S

Searches down (for a caller) frame whose function matches *Frame*. Sets point to the first stack-address (FP|0) of the frame (if found).

Frame {*string*} Searches for a frame whose function contains this string.

keywords :Detailed, :Reverse

Note that the IFEP Debugger shows all frames (including invisible and interpreter frames). Therefore, there is no need for the Lisp Debugger c-m-P or m-sh-P accelerators, or for the :Invisible or :Internal keywords.

:Detailed {Yes, No} Shows locals and disassembled code for the frame. (Default is No. Mentioned default is Yes.)

:Reverse {Yes, No} Searches up (for a callee) instead. (Default is No. Mentioned default is Yes.)

Key-binding accelerators

The accelerator prompts you for the *Frame* string to search for. Because of the way accelerators work, `c-- c-S` actually translates to the command :Reverse Find Frame, which is identical to :Find Frame with the sense of the :Reverse keyword reversed.

`c-S` :Find Frame

`c-- c-S` :Find Frame :Reverse Yes

Next Frame IFEP Debugger Command

:Next Frame *keywords* `c-N, m-N`

Shows the next frame (down) in the stack. Sets point to the first stack-address (FP|0) of the frame.

keywords :Detailed, :N Frames

:Detailed {Yes, No} Shows locals and disassembled code for the frame. (Default is No. Mentioned default is Yes.)

:N Frames {*base 10 integer*} Moves this many frames before displaying the frame. (Default is 1.)

Key-binding accelerators

Note that the IFEP Debugger shows all frames (including invisible and interpreter frames). Therefore, there is no need for the Lisp Debugger `c-m-P` or `m-sh-P` accelerators, or for the :Invisible or :Internal keywords.

`c-N` :Next Frame :N Frames <accelerator-argument>

`m-N` :Next Frame :N Frames <accelerator-argument> :Detailed Yes

Previous Frame IFEP Debugger Command

:Previous Frame *keywords* `c-P, m-P`

Shows the previous frame (up) in the stack. Sets point to the first stack-address (FP|0) of the frame.

keywords :Detailed, :N Frames

:Detailed — {Yes, No} Shows locals and disassembled code for the frame. (Default is No. Mentioned default is Yes.)

:N Frames — {*base 10 integer*} Moves this many frames before displaying the frame. (Default is 1.)

Key-binding accelerators

Note that the IFEP Debugger shows all frames (including invisible and interpreter frames). Therefore, there is no need for the Lisp Debugger c-m-P or m-sh-P accelerators, or for the :Invisible or :Internal keywords.

c-P — :Previous Frame :N Frames <accelerator-argument>

m-P — :Previous Frame :N Frames <accelerator-argument> :Detailed Yes

Top of Stack IFEP Debugger Command

:Top of Stack *keywords* m-<

Goes to the top of the stack (that is, the most recent frame). Sets point to the first stack-address (FP|0) of the frame.

keyword — :Detailed

:Detailed — {Yes, No} Shows locals and disassembled code for the frame. (Default is No. Mentioned default is Yes.)

Key-binding accelerators

m-< — :Top of Stack

Point Stack Accelerators

The IFEP Debugger provides two accelerators for maintaining the Point Stack (or PDL). They correspond closely to the point-pdl commands in Zmacs (c-SPACE for (m-X) Set Pop Mark, and c-m-SPACE for (m-X) Move to Previous Point). You can use these accelerators to save points of interest.

The Point Stack accelerators have no full-form equivalents in the IFEP Debugger command table. They are available only as accelerators.

The IFEP Debugger DDT commands automatically maintain the Point Stack by pushing the previous point on the stack whenever you move to a non-consecutive location.

When non-DDT commands set point, they do not always push the previous point on the Point Stack. Non-DDT commands only push the previous point onto the stack

if the location is still open when you enter the command. (You can usually return to a point that you reached with non-DDT Debugger commands by reissuing a single IFEP Debugger command.)

Since the Point Stack is a limited resource, oldest points are discarded when you add a point to a full stack.

The IFEP Debugger Point Stack accelerators are:

c-SPACE for **push-or-pop-point-pdl**.

With no argument, it pushes a point on the stack. This is equivalent to the Zmacs command (m-X) Set Pop Mark.

With an argument of c-U, it pops a location off the stack and sets point.

With an argument of c-U c-U, it pops a location of the stack and discards it.

With an argument of 0, it displays the Point Stack (describing each location on the stack, as if by :Describe Location).

c-m-SPACE for **exchange-point-pdl**

A numeric argument rotates top argument entries of the point PDL (the default numeric argument is 2). An argument of 1 rotates the whole point PDL, and a negative argument rotates the other way.

This is equivalent to the Zmacs command (m-X) Move Previous Command.

For either accelerator, if point is set from the stack, a :Show Location Contents command for the new value of point is executed.

Exchange Point IFEP Debugger Point Stack Accelerator

c-m-SPACE

IFEP Debugger Point Stack accelerators have no full-form command equivalents. They are available only as accelerators.

You can use this accelerator to exchange point with the top of the stack. (This is equivalent to the Zmacs command (m-X) Set Pop Mark.) A numeric accelerator allows you to rotate top argument entries of the Point Stack.

Note that, if point is set from the stack, :Show Location Contents for the new value of point is automatically executed.

Key-binding accelerators

c-m-SPACE	Exchanges point and the top of the Point Stack.
<*a number*> c-m SPACE	A numeric argument rotates top argument entries of the Point Stack. (Default numeric argument is 2). An argument of 1 rotates the whole Point Stack. A negative argument rotates the other way.

Push or Pop Point IFEP Debugger Point Stack Accelerator

c-SPACE

IFEP Debugger Point Stack accelerators have no full-form command equivalents. They are available only as accelerators.

You can use this accelerator to add or delete points from the IFEP Debugger Point Stack (or PDL). Accelerator options allow you to display the PDL, pop and discard a point from the PDL, pop and set a point, and push a point onto the Point Stack. This is equivalent to the Zmacs command (m-X) Move Previous Command.

Note that, if point is set from the stack, :Show Location Contents for the new value of point is automatically executed.

Key-binding accelerators

c-SPACE	Pushes a point onto the Point Stack.
c-U c-SPACE	Pops a location off the Point Stack and sets point.
c-U c-U c-SPACE	Pops a location off the Point Stack and discards it.
0 c-SPACE	Displays the Point Stack.

DDT Commands

The IFEP Debugger runs on the same processor as Lisp. This means that the IFEP Debugger has access to the Lisp structures. It also means that the IFEP Debugger and Lisp may be effected by the same error.

In addition to the commands that closely depend on the Lisp state, the IFEP Debugger provides a set of commands that only minimally depends on the Lisp state. These latter commands are especially useful when Lisp encounters an error. They are similar to DDT commands of conventional debuggers.

These commands provide basic debugging functions. You can use DDT commands to examine and modify Lisp memory. (Note that IFEP Debugger DDT commands use an address-oriented rather than object-oriented view of the Lisp world.) Using these commands requires knowledge of both hardware and software internals.

Almost all the IFEP Debugger DDT commands are technically accelerators, that is, they are single-character commands. Unlike the other accelerators, however, they do not normally echo their full-form equivalents (some of which are not even in the command table, for space reasons).

Describe Location IFEP Debugger Command

:Describe Location *Address keywords* =

Briefly describes what an address points to. If point is not open, the default argument is point. Otherwise, the default argument is the contents of point.

- For virtual addresses, the output of this command is similar to the output of **describe** for a locative (you could think of it as printing a *symbolic* address).
- For physical addresses that map to external bus addresses, the external bus address (and shuffling, if any) is given. For more information, see the macro **sys:with-bus-mode**, and see the macro **sys:with-hardware-bit-shuffling**.
- For addresses in other segments, the symbolic name (if any) is given.

Address
{*an address*} The address to describe. (Default is point.)
keywords
:Print Location, :Segment
:Print Location
{Yes, No} Specifies whether to echo the address and segment arguments first. (Default is No if point is open, Yes otherwise.)
:Segment
{*a segment*} The segment to interpret the address in. (Default is the segment of point.)

This command is most useful in its accelerator form (because of the way it accepts its arguments).

Note that the IFEP Debugger indicates that a location is open by *not* prompting for a new command. See the section "Using IFEP Debugger Commands".

For examples of how to use this command, see the section "Examples of Using the IFEP Debugger to Set Lisp Variables".

Key-binding accelerators

=	:Describe Location <accelerator argument> :Segment <argument modifier>

Set Location Contents IFEP Debugger Command

:Set Location Contents *Address Tag Pointer keywords* ^, RETURN, LINE, SPACE, BACK-SPACE

Sets the contents of a memory location.

Address	{*an address*} The address of the location whose contents to change. (Default is point.)
Tag	{*an integer in the current base between 0 and 255*} The Tag field of the value to set. (Defaults to the Tag field of point.)
Pointer	{*an address*} The pointer field of the value to set. (Defaults to the Pointer field of point.)
keywords	:Follow Forwarding, :Format, :Segment, :Then Show
:Follow Forwarding	{Yes, No} Specifies whether to follow any invisible pointers before storing the value. (Defaults to the current setting of the option as set by :Set Debugger Options.)
:Format	{*a typeout format*} The format to display the next location contents in. (Defaults to the default set by :Set Typeout Format.)
:Segment	{*a segment*} The segment to use for interpreting addresses. (Default is the segment of point.)
:Then Show	{*an address* or Null} A location to display next. (Default is Null.)

As with :Show Value in Format, this command is most useful in its accelerator form (because of the way the command accepts its arguments).

Note that these accelerators are only available when a location is open and an accelerator argument has been pressed. If a location is not open, or no accelerator argument has been pressed, these accelerators act as described in the :Show Location Contents IFEP Debugger command. (The RETURN accelerator with no argument simply closes the current location without modifying it.)

Key-binding accelerators

Because a typing mistake could easily be interpreted as a request to change location contents, the IFEP Debugger normally prompts for confirmation. If you expect to make a number of changes, you respond with P (for Proceed) to stop the Debugger from asking each time.

For examples of how to use these accelerators, see the section "Examples of Using the IFEP Debugger to Set Lisp Variables".

RETURN	:Set Location Contents <point> <accelerator-modifier> <accelerator-argument>
LINE, SPACE	:Set Location Contents <point> <accelerator-modifier> <accelerator-argument> :Then Show <point>+1
^, BACK-SPACE	:Set Location Contents <point> <accelerator-modifier> <accelerator-argument> :Then Show <point>-1

Set Typeout Format IFEP Debugger Command

:Set Typeout Format *Format*

Sets the default typeout format (that is, the format that IFEP Debugger DDT mode commands use to display their values). See the section "Using IFEP Debugger Typeout Formats".

You can use the :Show Value in Format IFEP Debugger command to redisplay the current value in another format.

This command is most useful in its accelerator form (because of the way it accepts its arguments).

Format {*a typeout format*} The format to use. (Defaults to the current setting.)

Not all the typeout formats are useful as defaults (for instance, C or Control Register). Common default typeout formats are:

- O "Octal"
- Q Lisp pointer
- S Lisp object

Show Location Contents IFEP Debugger Command

:Show Location Contents *Address keywords* /, LINE, SPACE, ^, BACK-SPACE, TAB

Shows the contents of a memory location.

This command is most useful in its accelerator form (because of the way it accepts its arguments).

Address {*an address*} The address of the location to display. (Defaults to point.)

keywords :Follow Forwarding, :Format, :Print Location, :Segment

:Follow Forwarding {Yes, No} Specifies whether to follow any invisible pointers before showing the value. (Defaults to the current setting of the option as set by :Set Debugger Options.)

:Format {*a typeout format*} The format of the location contents display. See the section "Using IFEP Debugger Typeout Formats". (Defaults to the default set by :Set Typeout Format.)

:Print Location {Yes, No} Specifies whether to echo the address and segment arguments first. (Default is Yes.)

:Segment {*a segment*} The segment to interpret the address in. (Default is the segment of point.)

Key-binding accelerators

The accelerators for this command are the main ways of examining Lisp memory.

/ Opens point and displays its contents. Alternatively, you can give an argument address and a segment.

LINE or SPACE Displays the next consecutive location (usually used without an argument).

^ or BACK-SPACE Displays the previous consecutive location (usually used without an argument).

TAB Goes *through* a location to display what that location addresses (usually used without an argument). Use this only if the current location actually contains an address.

When a location is open and you press an argument followed by any of LINE, SPACE, ^, or BACK-SPACE, the IFEP Debugger assumes a request to write the argument into the open location and then to display the location the accelerator would normally display. It does not treat the argument as an accelerator argument. For more information, see the section "Set Location Contents IFEP Debugger Command".

If you intend the argument as a new location to open, you must first close the currently open location (using RETURN). This returns you to the IFEP Debugger command prompt.

Use c-HELP or :Accelerator Help to display the accelerator descriptive names.

/ :Show Location Contents <accelerator argument> :Segment <argument modifier>

The default argument is point. (If an argument is given, :Print Location is No. Otherwise, the default is Yes.)

LINE, SPACE :Show Location Contents <accelerator argument> :Segment <argument modifier>

The default argument is point, incremented by one. If a location is open, an argument is interpreted as a :Set Location

Contents request. Otherwise, the argument is interpreted as a location and its contents is displayed.

^, BACK-SPACE — :Show Location Contents <accelerator argument> :Segment <argument modifier>

The default argument is point, decremented by one. If a location is open, an argument is interpreted as a :Set Location Contents request. Otherwise, the argument is interpreted as a location and its contents is displayed.

TAB — :Show Location Contents

The default arguments are the address and segment as indicated by the contents of point (that is, point is "indirected through" to find the location to display. (Note that, if an argument is given, it is taken as a location to "indirect through".)

See the section "Using Invisible Pointers or Cell Forwarding in the IFEP Debugger" for information on how to correctly use this accelerator.

Show Value in Format IFEP Debugger Command

:Show Value in Format *Tag Pointer keywords* ;, ESCAPE;

Displays a value in a specified typeout format. See the section "Using IFEP Debugger Typeout Formats".

Note that this command automatically follows cell forwarding before performing its function.

Tag — {*an integer in the current base between 0 and 255*} The tag field of the value to show. (Defaults to the tag field of point.)

Pointer — *{an address}* The pointer field of the value to show. (Defaults to the pointer field of point.)

keyword — :Format

:Format — {*a typeout format*} The typeout format to use. See the section "Using IFEP Debugger Typeout Formats". (Defaults to the default set by :Set Typeout Format.)

This command is most useful in its accelerator form (because of the way it accepts its arguments).

Key-binding accelerators

;*char* — :Show Value in Format <default> <default> :Format *char*

ESCAPE;*char* — :Show Value in Format <default> <default> :Format *char*

This command performs the same function as *;char*, except that it also sets the *temporary* default format to the format you specified. The *temporary* default applies for as long as you have an open location (that is, until you return to the IFEP Debugger prompt). See the section "Using IFEP Debugger Commands" for a definition of an open location.

For example, if you are examining a stack frame with the IFEP Debugger and see a display like this:

```
FOO:BAR (PC = 3)
Local 0:  <<Error printing value>>
→
```

You can change the typeout format by pressing `c-0 c-m-L` to get point to point to that stack slot, then enter `;Q` to see the value in Lisp pointer format.

```
FOO:BAR (PC = 3)
Local 0:  <<Error printing value>>
→:Show Local 0
<<Error printing value>>
→;q CDR-NEXT DTP-NULL 20012362724
→
```

Since Ivory allows random data-types on its stack, this might or might not be a legitimate value.

Note that the non-DDT IFEP Debugger commands all print out using S typeout format.

Lisp-like Commands

These commands provide debugging functions normally provided in Lisp.

Describe Area IFEP Debugger Command

:Describe Area *Area*

Describes an area (that is, performs **(**describe-area *"area"***)**.

In this example, the default is used. Note that since point is an address in a wired function, the default is Wired-control-tables (the area where wired functions are stored).

```
37001053101@V/17703377140013  .=Word 23 of #<DTP-COMPILED-FUNCTION
STORAGE:WIRED-WAIT 37001053056>
→Describe Area (Area [Default Wired-control-tables]) Wired-control-tables

Area #1: SYS:WIRED-CONTROL-TABLES has 1 region, region size 10000000 (octal):
   Last (dynamic) level: 0 regions, 0K allocated, 0K used.
   Static: 1 region, 64K allocated, 43K used.
       Region #1: Origin 37001000000, Length 200000, Free 125425, GC
125425, NEW STRUCTURE, Scav WIRED FixedSize NEW STRUCTURE Space: 1 region,
65536 allocated, 43797 used.
   Total for SYS:WIRED-CONTROL-TABLES: 65536 allocated, 43797 used.
→
```

Area {*an area name*} The name of the area to describe.

Describe Physical Address IFEP Debugger Command

:Describe Physical Address *Address*

Gives detailed virtual-memory information about a physical address. This command is used to debug virtual-memory errors.

For example,

```
→Describe Physical Address (Address [Default 37000000753]) 0
PMA 0 is in PPN 0
PPN 0 contains a copy of VPN 76000000
The load bit for VPN 76000000 is clear.
The sysout bit for VPN 76000000 is clear.
VPN 76000000 is in region 0 of area 0 (SI:FEP-AREA).  Region #0: Origin
37000000000, Length 1000000, Free 327642, GC 0, NEW LIST, NoScav WIRED FixedSize
PPN 0 is described by MMPT entry 0
  MMPT entry 0 describes VPN 76000000
  Status=WIRED, Write-Lock=0, Flushing=0, Wired-Count=7, Thread=77777777
```

Address {*an address*} The address to describe. (Defaults to point, as a physical address.)

Describe Region IFEP Debugger Command

:Describe Region *Region*

Describes a region (that is, performs **(si:describe-region** *si:region*).

Region {*a region number*} The region to describe. (Defaults to the region containing point.)

Describe Virtual Address IFEP Debugger Command

:Describe Virtual Address *Address*

Gives detailed virtual-memory information about a virtual address. This command is used to debug virtual-memory errors.

For example,

```
→Describe Virtual Address (Address [Default 37000000753]) 0
The load file contains a copy of VPN 0 at DPN 561555, in a block of 1 pages at DPN 561555.
The load bit for VPN 0 is set.
The sysout bit for VPN 0 is clear.
VPN 0 is in region 112 of area 10 (SYS:WORKING-STORAGE-AREA).
        Region #112: Origin 0, Length 1000000, Free 220, GC 0, NEW LIST,
Scav EPHEMERAL level 1
```

Address {*an address*} The address to describe. Defaults to point, as a virtual address.

Symbol Function IFEP Debugger Command

:Symbol Function *Symbol*

Shows the definition of a Lisp symbol (that is, performs **(symbol-function** *'symbol*)).

Note that the IFEP Debugger always shows the top-level binding at the time Lisp halted. Also note that it shows special variable bindings for stack-frames that are displayed (it does *not* look up bindings for a particular stack frame or process).

Sets point to the function-cell of the symbol.

Symbol {*an interned Lisp symbol*} The symbol whose definition to show.

Symbol Value IFEP Debugger Command

:Symbol Value *Symbol*

Shows the value of a Lisp symbol (that is, performs **(symbol-value** *'symbol*)). You can use this command to see the current value of a special variable.

Note that the IFEP Debugger always shows the top-level binding at the time Lisp halted. Also note that it shows special variable bindings when stack frames are displayed (it does *not* look up bindings relative to a particular stack-frame or process.)

Sets point to the value-cell of the symbol.

Symbol {*an interned Lisp symbol*} The symbol whose value to show.

Miscellaneous Commands

These commands perform functions that do not fit neatly into the other command categories.

Abort IFEP Debugger Command

:Abort c-Z, ABORT

Exits the IFEP Debugger.

Note that if you are not at the top-level Debugger prompt, ABORT may have other meanings (such as, terminate the current typein, typeout, or command in execution). You may need to enter ABORT several times to achieve the same effect as :Abort (that is, to exit from the IFEP Debugger).

Key-binding accelerators

c-Z, ABORT	:Abort

Accelerator Help IFEP Debugger Command

:Accelerator Help c-HELP

Gives a brief paragraph describing accelerator arguments and modifiers, then lists the accelerator keys and their descriptive names.

Key-binding accelerators

c-HELP	:Accelerator Help

Debug Process IFEP Debugger Command

:Debug Process *Process keyword*

Selects a process other than the current one. Use :Debug Process HELP to see a list of processes and their states.

Process *{a process name and address}* To uniquely identify the process, include the address of the process with the name. You can use command completion to include the process address. (Defaults to the process that was running when Lisp stopped.)

For a list of possible processes and their states, use HELP, c-/, or c-?.

keyword :Show Initial Frame

:Show Initial Frame {Yes, No} Specifies whether or not to show the top frame of the process.

Use :Show Initial Frame No if it appears that something in the initial display is preventing the Debugger from debugging the process. (Default is Yes. Mentioned default is No.)

Mail Bug Report IFEP Debugger Command

:Mail Bug Report *Nframes keyword* c-M

Saves a backtrace that you can later include in Lisp bug mail.

Nframes {*base 10 integer*} How many frames to include in the backtrace. (Default is 5.)

keyword :Output Destination

:Output Destination {*FEP pathname or* Null} The FEP saves the backtrace in local memory, unless you name a destination disk file. Note that FEP local memory is limited (and ephemeral, if the FEP is reset). (Default is Null. Mentioned default is FEP:>Crash-data.text.newest.)

Key-binding accelerators

c-M	:Mail Bug Report <accelerator-argument>
ESCAPE c-M	:Mail Bug Report <accelerator-argument> :Output Destination

Set Base IFEP Debugger Command

:Set Base *Number Base*

Sets the default base for printing and accepting numbers in the IFEP Debugger.

Note that numeric arguments to command accelerators prefixed by any of c-, m-, s-, or h- are *always* parsed in base 10. Unprefixed numbers are parsed in the current base.

The base setting applies only for the current session of the IFEP Debugger. It is reset when you leave the Debugger. You cannot set input and output bases separately.

Number Base {*base 10 integer*} The number base to use. (Default is 10.)

Set Debugger Options IFEP Debugger Command

:Set Debugger Options *keywords*

Sets the values of various Debugger parameters.

keywords :Follow Forwarding, :Name Heuristication, :Print Errors, :Print Length, :Print Level

:Follow Forwarding {Yes, No} Specifies whether to hide invisible pointers as they are in Lisp. (Default is No.)

:Name Heuristication {Yes, No} Specifies whether to search for a plausible name when printing named structures and instances. (Default is Yes.)

:Print Errors {Yes, No} Specifies whether or not to print internal FEP debugging information, when the printer gets an error. (Default is No.)

:Print Level {*an integer*} The depth to abbreviate lists to when describing stack frames. Note that there is no abbreviation when examining individual objects. (Default is 2.)

:Print Length {*an integer*} The length to abbreviate lists to when describing stack frames. Note that there is no abbreviation when examining individual objects. (Default is 4.)

Set Default Segment IFEP Debugger Command

:Set Default Segment *Segment*

Specifies the default segment for interpreting an address.

Accelerator arguments that are entered without an argument modifier that explicitly specifies the segment, are considered to be in the default segment (as set by this command). The self-documentation for this command's argument gives the possible segments and their abbreviations.

Segment {*a segment*} The segment to use as a default for interpreting addresses. See the section "IFEP Debugger Accelerator Arguments" for a description of available segments. (Default is the current setting, initially V.)

Set Package IFEP Debugger Command

:Set Package *New Package*

Sets the default package for remote symbol lookup.

Note that the IFEP Debugger always prints symbol packages, even for symbols in the current package. Also note that the IFEP Debugger does not distinguish internal from external symbols. Therefore, do not interpret a internal symbol printed with a single colon to mean that its package is current. See the section "Printed Representation of Lisp Objects in the IFEP Debugger".

New Package {*a package*} The package to use. (Default is the current value of ***package***.)

Printed Representation of Lisp Objects in the IFEP Debugger

The IFEP Debugger implements the Lisp printer, with the following restrictions:

- The IFEP Debugger printer does not always print *prettily*.
- Long printed representations simply wrap. You can abort long printouts by pressing any key.
- The IFEP Debugger does not implement printing of circular objects. You must manually abort the printout.
- When describing a stack frame, ***print-level*** and ***print-length*** limits are 2 and 4, respectively. Otherwise, they are unlimited.

Lisp objects are printed in a familiar format, with the following exceptions:

- Symbols
- Numbers that are *not* fixnums
- Characters or strings with character-set or character-style attributes
- Bit vectors, vectors, and arrays
- Named-structures and instances

The printed representation describes the S typeout format. For more details, see the section "Using IFEP Debugger Typeout Formats".

Printing Symbols

The IFEP Debugger does not print symbols in the same way that the Lisp printer does.

- The IFEP Debugger always prints the full package name of symbols (regardless of the current package, syntax, or existence of relative package names).

- The IFEP Debugger always prints one colon separating the package name from the symbol name (that is, the printer does *not* distinguish between internal and external symbols).

- The IFEP Debugger does *not* quote package or symbol names.

Printing Numbers

The IFEP Debugger prints fixnums in the current base. All other numbers print as their data-type and pointer field. For example, 1.0 prints as

```
#<DTP-SINGLE-FLOAT 7740000000>
```

You can change the number base using the :Set Base IFEP Debugger command.

Printing Characters and Strings

String-chars and strings of string-chars (also known as *thin* strings) print normally in the IFEP Debugger. Characters with only bits attributes also print normally.

Characters that have a non-nil character style or a non-zero character set attribute print as if thin, that is, as if their character-style and character set are ignored. If the thin character does not map into the FEP's only font, it is printed as a lozenged octal number instead. (This may have obscure results for fonts that do not represent standard characters. For example, **#\mouse:nw-arrow** prints as `<406>`.

Note that usually errors that require using the IFEP Debugger do not involve characters or strings in nonstandard fonts.

Printing Bit Vectors, Vectors, and Arrays

These objects are printed as arrays with print-array set to **nil** You can use the IFEP Debugger DDT commands to examine individual array elements. See the section "IFEP Debugger DDT Commands".

Printing Named Structures and Instances

The IFEP Debugger does not support print functions or methods for named structures and instances. However, it does include a heuristic that searches for and prints a string could plausibly be the name of the structure or instance. For example,

```
→Symbol Value *current-process*
#<PROCESS:PROCESS Zmacs Windows 20120021303>
→
```

This is similar to the Lisp printed representation,

```
#<PROCESS:PROCESS Zmacs Windows (2 1) 20120021303>
```

The heuristic looks for a string in the first few (15) slots of an instance or named structure (preferring leader slots, if the structure has a leader). If a string is found, it is printed in the "name" position, when the object is printed. For example,

```
#<TypeName "name" 000000>
```

This heuristic does not work for structures or instances whose name is not a string.

You can disable this heuristic (if it appears to be preventing the printer from working) by using:

```
:Set Debugger Options :Name Heuristication No
```

See the section "Set Debugger Options IFEP Debugger Command".

Printing Errors

When the IFEP Debugger printer gets an error, it tries a fall-back printer that simply prints the data-type and pointer field of the object. For example,

```
(<<Error printing #<DTP-LIST 2002232132>>>
```

You can use :Set Debugger Options :Print Errors Yes to get more information about the error. (For more information, see the section "Set Debugger Options IFEP Debugger Command".)

This is particularly useful when the object also seems to the be the source of the Lisp error. For example, with :Print Errors Yes, the previous example would print,

```
(<<Error MEMORY-DATA-ERROR referencing 2002232142 at #<DTP-ODD-PC
(543 in FEP::DEFAULT-PRIN1-PRINC) 3700021447> printing #<DTP-LIST 2002232132>>>
```

You can use the :Show Value in Format IFEP Debugger command to display objects in other (lower-level) representations to examine objects which may be causing IFEP Debugger printing problem. See the section "Show Value in Format IFEP Debugger Command".

Using IFEP Debugger Typeout Formats

IFEP Debugger commands are used mainly to examine locations. Many of the IFEP Debugger commands allow you to explicitly control how the value appears.

You can use the :Format keyword option of any of the following IFEP Debugger commands to specify a typeout format for value. (You can use the ; accelerator for :Show Value in Format to redisplay the last value shown in a different format.)

:Describe Location

:Set Location Contents

:Set Typeout Format

:Show Location Contents

:Show Value in Format

See the section "IFEP Debugger Command Descriptions" for more details.

Unless otherwise specified, values are displayed as 40-bit numbers in the current base. Since the default number base is 8, the default typeout format is "Octal".

Typeout formats are represented in IFEP Debugger DDT commands by single characters. The possible typeout formats are:

' Characters. The 32-bit pointer field of the value is printed both as four comma-separated string-chars and as a single (parenthesized) character. The data-type and cdr-code of the value are ignored.

For example, in ' typeout format

```
00400023044516 is N, I, L, • (<44516>)
```

and

```
02162000000204 is <FUNCTION>, •, •, <220> (h-c-FUNCTION)
```

Bit Numbers. The value is treated as a 40-bit word, and is printed as a comma-separated list of the bit numbers (base 10) of the bits that are on (or 1) in the word.

For example, in # typeout format

```
16360043371350 is 3,5-7,9,12-16,18,19,23,31-34,37-39
```

A Array Header. The value must be of type **sys:dtp-header-i.** It is interpreted as an Ivory array header. You can use this format to examine an array when that array is damaged in a way that prevents the IFEP Debugger printer from decoding it.

C Control Register. The value must be a fixnum. It is interpreted as an Ivory control register and its various subfields printed out symbolically. You can use this format to decode a stack frame when that stack or frame is damaged in a way that prevents the IFEP Debugger Stack Frame Display commands from parsing it.

E Error Trap. The value must be a fixnum. It is looked up in the error-trap dispatch table and the corresponding entry printed. (This is somewhat analogous to the **dbg:decode-micro-pc** Lisp function).

I Instructions. The value is interpreted as an Ivory instruction and its symbolic representation is displayed. The Debugger attempts to find the

enclosing function and to display appropriate relative PC(s). If the PCs are negative, the Debugger failed to find the function header. You can use this format to examine instructions in a function when that function is damaged in a way that prevents the :Show Compiled Code IFEP Debugger command from working.

O "Octal". The value is treated as a 40-bit integer and printed in the current base. Note that this is octal *only if* the current base is 8 (the default).

Q Lisp Pointer. The value is decomposed into a cdr-code data-type and pointer field. The cdr-code and data-type are printed symbolically followed by the pointer field as an address (unsigned 32-bit integer).

S Lisp Object. The value is treated as a Lisp object and is printed as if by **print** (the mnemonic S is named after the ~**s** format directive of **format**). Note that this format automatically follows cell-forwarding before printing.

More About Using the IFEP Debugger

This section includes other useful information about using the IFEP Debugger.

The IFEP Debugger and Virtual Memory

The IFEP Debugger has access to Lisp virtual memory. When examining virtual memory, the IFEP Debugger uses resident and pending pages first (possibly completing a disk read that Lisp had started). If the page is not resident or pending, the FEP swaps it into FEP space. (Note that because the FEP paging space is limited, swapping may appear to be slow.)

If Lisp crashed because of a disk error in its swap image, the IFEP Debugger will not be able to examine the page in error (unless the disk error is first remedied). Instead the IFEP creates a page of NULLs (with an appropriate error message).

If the IFEP Debugger gets an error trying to set up its virtual memory system, or if you enter the IFEP Debugger with :Debug :Ignore Storage Structures Yes (see the section "Entering the IFEP Debugger"), the Debugger assumes that Lisp has not initialized its storage system, and attempts to proceed without relying on Lisp structures (such as the load-bitmap, address-space-map, or swap-map).

In this case, the IFEP Debugger pages in using the load-map. Since the Debugger assumes that storage was never initialized, you will not see modified pages that are in the swap-map.

Using Invisible Pointers or Cell Forwarding in the IFEP Debugger

The Ivory architecture supports cell forwarding or *invisible* pointers. In the IFEP Debugger, invisible pointers are normally *not* invisible. That is, by default, cell forwarding is disabled in the IFEP Debugger.

If you do not need to examine invisible pointers, you can enable automatic cell forwarding by using the IFEP Debugger command :Set Debugger Options. See the section "Set Debugger Options IFEP Debugger Command".

If you need to examine invisible pointers using the IFEP Debugger, note that the TAB accelerator does *not* automatically follow cell forwarding. So, you may need to use several TAB commands to achieve the same result as a single **location-contents.**

The S typeout format and the Stack Frame display commands are exceptions to the rule (see the section "IFEP Debugger DDT Commands"). Each automatically follows cell forwarding before printing.

Examples of Using the IFEP Debugger to Set Lisp Variables

Because the IFEP Debugger has access to Lisp, you can use Debugger commands to set the value of Lisp variables when you are unable to do so from Lisp, perhaps because an error is occurring early in booting. For example, you can use the IFEP Debugger to turn on a level of debugging, or to disable some operation that is failing.

Note that when you use the IFEP Debugger to set Lisp variables, you must be familiar with the system architecture and the internal representation of Lisp structures.

Example

This example uses the IFEP Debugger to set the Lisp variable **rpc:*server-debug-flag*** to diagnose an error during warm-booting (before Lisp is initiated).

The :Symbol Value IFEP Debugger command sets point to the location of the value-cell of the Lisp symbol. DDT commands on point are then used to modify that value.

Use h-c-FUNCTION to get to the FEP, and :Debug to enter the IFEP Debugger. Then use the :Symbol Value IFEP Debugger command to see the value of the Lisp symbol **rpc:*server-debug-flag***. Note that this command sets point to the location of the value-cell of the symbol.

```
Debugger Command: Symbol Value (Symbol) rpc::*server-debug-flag*
:NOTIFY
→
```

There are several ways to display the value. Press / to open point.

```
→20012362724@V/00260013616051
```

Press ESCAPE;q to set the typeout format to Q.

```
→20012362724@V/00260013616051 ESCAPE;q CDR-NEXT DTP-ONE-Q-FORWARD 20013616051
```

This shows that the value cell for this symbol is *forwarded*. The IFEP Debugger does not automatically follow forwarding. You can explicitly follow forwarding by pressing TAB.

```
→20012362724@V/00260013616051 ESCAPE;q CDR-NEXT DTP-ONE-Q-FORWARD 20013616051
→20013661051@V/CDR-NEXT DTP-SYMBOL 20013043610
```

Since there is no more forwarding, you can verify that you are in the right place by pressing ;s.

```
→20012362724@V/00260013616051 ESCAPE;q CDR-NEXT DTP-ONE-Q-FORWARD 20013616051
→20013661051@V/CDR-NEXT DTP-SYMBOL 20013043610 ;s :NOTIFY
```

Since the location is open, you can change its value by pressing 't RETURN.

```
→20012362724@V/00260013616051 ESCAPE;q CDR-NEXT DTP-ONE-Q-FORWARD 20013616051
→20013661051@V/CDR-NEXT DTP-SYMBOL 20013043610 ;s :NOTIFY (symbol) 't
Do you really want to write 01437001011010 into 20013661051@V? (Y, N, or P)
```

When you type ' with a location open, the Debugger prompts with "(symbol)". The debugger expects a value to write, rather than a location. (At the top level, pressing a symbol or ' prompts with "Examine (Symbol).) When you press RETURN, the Debugger asks if you want to write into the open location (instead of just closing the location). Answer Y, and then check your work.

```
→20012362724@V/00260013616051 ESCAPE;q CDR-NEXT DTP-ONE-Q-FORWARD 20013616051
→20013661051@V/CDR-NEXT DTP-SYMBOL 20013043610 ;s :NOTIFY (symbol) 't
Do you really want to write 01437001011010 into 20013661051@V? (Y, N, or P) y
→Debugger Command: Symbol Value (symbol) rpc::*server-debug-flag* T
→
```

Press :Start to rerun the warm-booting process. The next time an RPC Server error occurs, the Debugger will be entered (in the cold-load stream).

Using a Serial Terminal to Communicate with the FEP

You can use a serial terminal to communicate with the FEP. One case in which this can be particularly useful is in troubleshooting an XL1200 single-monitor color station. For example, if the color monitor cannot come up, you can connect a serial terminal to the serial port on the color console unit, and use that terminal to give FEP commands such as Show Disk Label, or Set FEP Options.

In addition to connecting the serial terminal physically, you need to give the Set Console FEP command to tell the FEP to use the serial console. You might need to also use the Set Monitor Type FEP command to tell the FEP whether the serial console is ASCII (a dumb terminal) or X3.64 (such as a VT100).

You need to set the serial terminal's parameters for 9600 baud, 8 bits, and no parity.

Keep in mind that serial terminals don't have all the special keys of the Symbolics keyboard. If you need to transmit special Symbolics characters, such as Meta or Super characters, you need to understand how serial terminal keys are mapped to the Symbolics keys.

Mapping of Serial Terminal Keys to Symbolics Keys

The keyboard of a serial terminal does not have the same set of keys as does the Symbolics keyboard. For example, such a keyboard typically lacks a Meta key, Super key, Hyper key, and Symbol key. These keyboards do, however, have a Control key, however, most such keyboards can handle only Control-A, Control-Z, and a few other characters, most of which are reserved for escapes.

Accessing the Symbolics Character Set

The following characters may be used to access the Symbolics character set:

```
c-^ = Toggles the Control bit   c-] = Toggles the Super bit
ESC = Toggles the Meta bit      c-\ = Toggles the Hyper bit
c-@ = Toggles the Shift bit
```

For example, to enter c-m-C, you need to set both the Control bit and the Meta bit, by entering c-^ and ESC; you can then press C to enter c-m-C.

Similarly, you might need to enter c-sh-C. The serial keyboard has both a Control key and a Shift key, but you cannot press them both at once to enter c-sh-C. You can enter c-^ to set the Control bit, then press the Shift key while typing C. Or, you can enter c-@ to set the Shift bit, and then press the Control key while typing C.

Entering Special Symbolics Keys

The character c-_ (that is, the Control key and the underscore _ key) is used as a prefix to enter special characters as follows:

```
H = <Help>                  L = <Line>
E = <End>                   P = <Page>
A = <Abort>                 F = <Refresh>
S = <Suspend>               B = <Back-Space>
R = <Resume>                N = <Network>
C = <Complete>              1 = <Square>
I = <Clear-Input>           2 = <Circle>
X = <Escape>                3 = <Triangle>
```

For example, if you press c-_ followed by H (that is, two keystrokes) on the keyboard of a serial terminal, you get the effect of the HELP key on a Symbolics keyboard.

Entering Symbol Characters

c-_ _ is the prefix for Symbol characters. (That is, Control Underscore followed by Underscore, two keystrokes.)

For example, you can enter c-_ _ P (that is, three keystrokes) to get the effect of SYMBOL-P.

c-_ ? displays the c-_ dispatch table.

The Lisp Machine File System (LMFS)

All Symbolics computers use Lisp Machine File System (LMFS) files and directories. This section provides information about utilizing your disk to maximize the space available for LMFS files (by using LMFS partitions). This section also describes the procedures for backing up, dumping, reloading, and retrieving files (and systems) in LMFS.

Introduction to LMFS

The Lisp Machine File System (LMFS) provides a file system that runs on a Symbolics computer and stores information on the computer's disks. The information can be accessed locally, or from other computers.

LMFS Concepts

LMFS files are categorized as character files and binary files. Character files consist of a certain number of characters (the *byte count*) in the Symbolics character set. Binary files consist of a certain number of binary data bytes (unsigned binary numbers up to sixteen bits in length).

LMFS files have a *name*, a *type*, and a *version*. The name is a character string of any length. The type is a character string of up to fourteen characters in length. The version is a positive integer up to 16777215. Names and types can consist of upper and lower case characters (when searching for file names, the system is not case-sensitive). If you create a file whose name is Foofile, you can ask for foofile, FOOFILE, or FOOfile. Neither the greater-than character [>] nor a RETURN can appear in filenames or types.

Each filename is an arbitrary user-chosen string describing the file. The type should indicate what type of data the file contains; a file of type lisp should contain Lisp source programs, and a file of type bin should contain compiled Lisp programs, for example. The version number denotes successive generations of a file. One way to create a new version of a file is to read in the latest version of the file, modify it, and write it out. For more information:

- See the function **open**.
- See the function **with-open-file**.
- See the section "Naming of Files".

Files reside in *directories*. The combination of name, type, and version of any file is unique within the directory where it resides. With the exception of a single directory (the *root*), directories can reside in other directories. The directory in which a file or directory resides is called its *parent* directory. Files and directories are said to be *inferior* to their parent directories.

Directories and files thus form a strict tree (hierarchy); the root directory is the root of the tree. Directories are of type "directory" and have a version of 1. Thus,

the name of a directory alone identifies it within its containing (*superior*) directory. It is not possible to "fool" LMFS into thinking a file is a directory by giving it a type of "directory" and a version of 1, however.

Links are the third (and last) kind of object that can reside in a directory. A link contains the character-string representation of the pathname of something else in the same file system, called the *target* of the link. This pathname specifies a directory, a name and a type, and it can specify a version. A link is conceptually an indirect pointer to something else; when certain operations are performed on a link, the operation really gets performed on the target of that link.

It is possible to have directory links. See the section "LMFS Links".

The specific syntactic conventions, restrictions, and other information about LMFS pathnames are described elsewhere. See the section "LMFS Pathnames".

LMFS also stores and maintains *properties* of objects. For example, for each file, LMFS stores the creation date, the author, and information about whether the file has been backed up. Users can also create their own properties for objects; each file has a property list that lets you store arbitrary information associated with the file. See the section "LMFS Properties".

The File System Maintenance Program contains the File System Editor (FSEdit), an interactive program that lets you manipulate the local LMFS system or the system on any host. Type `SELECT F` to invoke the program and to use FSEdit. For information about FSEdit, see the section "Using FSEdit".

Before you use the file system on a machine, log in to it. If you are using the file system locally, log out of the machine before you cold boot it (especially if you have created files on the file system or expunged directories while using it). Otherwise, you will run out free disk records (at the rate of about 30 to 50 records for each boot session in which files were created), and the free record salvager will have to be run.

LMFS Properties

Files, directories, and links have various *properties*. There are *system* properties (defined and maintained by the file system itself), and *user* properties (defined and maintained by programs and people that use the file system). Every property has a *name*, which is a keyword symbol, and a *value*, which is a Lisp object.

This section contains a listing of the names of all of the system properties.

Note: System properties should not be confused with the *file attribute list* (the **-*-** line in the beginning of a file). For information about the **-*-** line, see the section "File Attribute Lists".

You can examine the values of properties by using either the [Show Properties] command in the File System Editor, or Show File Properties (`m-X`) in Zmacs. Users alter the values by using the either the [Edit Properties] command in the File System Editor, or Change File Properties (`m-X`) in Zmacs. For information about FSEdit, see the section "Using FSEdit".

Programs access the values of properties by using the **fs:directory-list** and **fs:file-properties** functions and alter the values by using the **fs:change-file-properties** function. See the section "Accessing Directories".

Some system properties apply to files, directories, and links alike. For example, all these objects have an *author* and a *creation time*. Other system properties are not defined for all kinds of objects. For example, only files have a *length-in-bytes* property, only directories have an *auto-expunge interval* property, and only links have a *link-to* property. Table 2 tells you which kinds of objects to which each property applies. (User properties can always apply to any object.)

The values of some system properties are determined by the file system and cannot be set by the user. For example, you cannot set the *length-in-bytes* nor the *byte-size*. The values of other properties can be changed arbitrarily. For example, you can set the *generation-retention-count* or the *don't delete* property whenever you want to do so. The properties of the latter set are called *changeable* properties. Properties in the first group reflect facts about the file, whereas those in the second group represent the current state of user-settable options regarding the file.

When the **fs:change-file-properties** function is called for a changeable system property, the property is changed. When it is called for a non-changeable system property, an error is signalled. When it is called for any property name that is not the name of one of the system properties in Table 1, it assumes that it is the name of a user property, and the property is established or changed.

When the **fs:file-properties** function is called for a LMFS file, it returns a second value: a list of the names of all the properties of the file that are changeable. This function lists all the system properties and all the user properties for the object it is given.

The names of user properties must be symbols on the keyword package, and must not be the same as any of the system property names. The value associated with a user property does not have to be a string, but it wil be converted to one after it is entered. The combined length of the name of the property and its value must not exceed 512 characters.

To remove a user property from a file, set the value of the property to **nil**. **fs:file-properties** returns all the user properties of a file, but **fs:directory-list** does not return any of them. You can create new user properties with the [New Property] command in the File System Editor; after they are created, you can edit them with [Edit Properties]. Programs create and change user properties by using **fs:change-file-properties**.

Note: If you misspell or otherwise misconstrue the name of a system property, LMFS will assume that you have given the name of a user property (and will set it). Because of this limitation, LMFS can never diagnose an unrecognized, or invalid, property name.

Table 2 lists all of the standard properties that LMFS maintains. The standard, generic interpretation and representations of the system standard properties among them can be found elsewhere: See the section "Functions for Accessing Directories".

```
:length-in-bytes
:byte-size
:author
:creation-date
:modification-date
:reference-date
:deleted
:not-backed-up
:dont-delete
:dont-reap
:open-for-writing (LMFS-specific)
:length-in-blocks
:generation-retention-count
:directory
:auto-expunge-interval
:date-last-expunged
:default-generation-retention-count
:default-link-transparencies (LMFS-specific)
:link-to
:link-transparencies (LMFS-specific)
:partition
:volume-name
:lmfs-directory-acl (if ACLs have ever been assigned to the directory)
```

Table 5. System Properties

The following among them are changeable, that is, users can set their values by means of **fs:change-file-properties**. These also appear in FSEdit if you use the [Show Properties] command:

```
:generation-retention-count
:modification-date
:reference-date
:creation-date
:author
:deleted
:dont-reap
:dont-delete
:dont-delete-reason
:auto-expunge-interval
:default-generation-retention-count
:default-link-transparencies
:link-transparencies
```

Table 6. LMFS System Properties

The following is a list of all the properties supported by LMFS that are either specific to LMFS or require other special comment.

:byte-size *(Files)*

For a character file, **8**. For a binary file, the byte size of the file (the number of bits in each byte), a fixnum between **1** and **16**, inclusive. LMFS maintains both the byte size and the binary/character quality of a file natively. It is not permitted to open a binary file with a byte size other than that with which it was written. This property is not currently a changeable one.

:length-in-blocks *(Files, directories, links)*

A *LMFS record* is 1152 32-bit words for 3600 family machines. For Ivory-based machines, a *LMFS* record is 1280 32-bit words. This is the basic allocation unit of the file system. The name of the generic system property is confusing in the case of LMFS, for a LMFS record is composed of multiple disk blocks. This property cannot be meaningful for directories.

:creation-date *(Files, directories, links) (Changeable)*

LMFS allows setting of creation date by user programs. Creation date, when not set by a user program, is also updated when a file is appended to.

:modification-date *(Files) (Changeable)*

The most recent time at which this file was modified, expressed in Universal Time. This is the same as the creation date if the file has been opened for appending. Operations such as renaming and property changing update this property, but do not update creation date. The dumper, for instance, is driven off this property.

:author *(Files, directories, links) (Changeable)*
This property is user-settable in LMFS.

:dont-delete *(Files, directories, links) (Changeable)*
If **t**, does not allow this object to be deleted. The purpose of this attribute is to prevent the accidental deletion of important files. An error results if an attempt is made to delete this file.

:dont-reap *(Files, directories, links) (Changeable)*
This attribute, although maintained internally by LMFS, is not interpreted by LMFS. Instead, Dired directory maintenance tools use this property.

:default-generation-retention-count *(Directories) (Changeable)*
The default value for the **:generation-retention-count** property of new objects created in this directory. **:generation-retention-count** is used by LMFS to control the number of versions of each file. See the section "Deletion, Expunging, and Versions in LMFS". The value should be **nil** or a nonnegative fixnum.

:auto-expunge-interval *(Directories) (Changeable)*
LMFS automatically expunges a directory whenever a file system operation is performed, provided the directory in question has this property and that amount of time has expired since the last time the directory was expunged, whether this previous expunging happened manually, or as a result of **:auto-expunge-interval**. All deleted files are expunged, and the time of their deletion is not taken into consideration. See the section "Deletion, Expunging, and Versions in LMFS". The value should be **nil** or a nonnegative fixnum.

:default-link-transparencies *(Directories) (Changeable)*
The initial value for the **:link-transparencies** attribute of links created in this directory. See the section "LMFS Links". To set this property, use the [Link Transparencies] command in the File System Editor rather than [Edit Properties].

:link-transparencies *(Links) (Changeable)*
The transparencies of this link. See the section "LMFS Links". To set this property, use either the [Edit Link Transparencies] or [Edit Properties] commands in the File System Editor, or Change File Properties (m-X) in Zmacs.

:complete-dump-date *(Files, directories, links)*
The most recent time at which this object was dumped on a complete dump tape, expressed in Universal Time. See the section "Dumping, Reloading, and Retrieving LMFS Files". A positive bignum. If this object has never been dumped on a complete dump tape, this property is not present. This property does not appear in directory listings.

:complete-dump-tape *(Files, directories, links)*
The tape reel ID of the complete dump tape on which this object was most recently dumped. A string. If this object has never been dumped on a complete dump tape, this property is not present. This property does not appear in directory listings.

:incremental-dump-date *(Files, directories, links)*

The most recent time at which this object was dumped on an incremental or consolidated dump tape, expressed in Universal Time. A positive bignum. If this object has never been dumped on an incremental dump tape, this property is not present. This property does not appear in directory listings.

:incremental-dump-tape *(Files, directories, links)*

The tape reel ID of the incremental or consolidated dump tape on which this object was most recently dumped. A string. If this object has never been dumped on an incremental dump tape, this property is not present. This property does not appear in directory listings.

Deletion, Expunging, and Versions in LMFS

When an object (file, directory, or link) in LMFS is deleted, it does not really cease to exist. Instead, it is marked as "deleted" and continues to reside in the directory. If you change your mind about whether the file should be deleted, you can *undelete* the file, which will bring it back.

The deleted objects in a directory go away when the directory is *expunged*; this can happen by explicit user command or by means of the auto-expunge feature (see below). When a directory is expunged, the objects in it disappear, and they cannot be brought back (except from backup tapes.) See the section "Finding Backup Copies of LMFS Files".

When a file is deleted, any attempts to open it will fail (as if the file did not exist). It is possible to open a deleted file by supplying the **:deleted** keyword to **open**, but this is rarely done.

Users normally delete and undelete objects with the Zmacs commands Delete File (m-X) and Undelete File (m-X), or [Delete] and [Undelete] commands in the File System editor, or D and U in Dired. Directories can be expunged with Dired or the File System Editor, with the Expunge Directory (m-X) command in Zmacs, or the Command Processor (CP) Expunge Directory command.

Programs normally delete files using the **delete-file** function. See the function **delete-file**. Whether a file is deleted or not appears as the **:deleted** property of the file, and programs can delete or undelete files by using **fs:change-file-properties** to set this property to **t** or **nil**.

Directories can optionally be automatically expunged. Every directory has an **:auto-expunge-interval** property, whose value is a time interval. If any file system operation is performed on a directory and the time since the last expunging of the directory is greater than this interval, the directory is immediately expunged. The default value for this property is **nil**, meaning that the directory should never be automatically expunged.

The normal way of writing files in the Genera environment is to create a new version of the file each time it is written. When you edit with Zmacs, for example, every time you use the Save File command a new version of the file is written out. After a while, you end up with many versions of the same file (but this clutters

your directory and uses up disk space). Zmacs has some convenient commands that make it easy to identify and automatically delete the old versions.

LMFS also has a feature that deletes the old versions automatically. A file property called the *generation retention count* says how many generations (that is, new versions) of a file should be kept. Suppose the generation retention count of a file is three, and versions 12, 13, and 14 exist. If you write out a new version of the file, then version 12 will be deleted, and now versions 13, 14, and 15 will exist. (Actually, version 12 is only deleted and not expunged, so you can still get it back by undeleting it.) If the generation retention count is zero, that means that no automatic deletion should take place.

The above explanation is simplified. You might wonder what would have happened if versions 2, 3, and 14 existed, and what might have happened if the different versions of the file had different generation retention counts. To be more exact: each file has its own generation retention count. When you create a new version of a file and some other version of the file already exists (that is, another file in the directory with the same name and type but some other version), the new file's generation retention count is set to the generation retention count of the highest existing version of the file. If there is no other version of the file, it is set from the *default generation retention count* of the directory. (When a new directory is created, its default generation retention count is zero, no automatic deletion.)

If you want to change the generation retention count of a file, you should change the count of the highest-numbered version; new versions will inherit the new value. When the new file is closed, if the generation retention count is not zero, all versions of the file with a number less than or equal to the version number of the new file minus the generation retention count will be deleted.

When a file version is being created, it is marked with the property **:open-for-writing**. This property is removed when the file is successfully closed. While the file has this property, it is invisible to normal directory operations and to attempts to open or list it.

Directory list operations that specify **:deleted** can see the file. Files in this state have the "open for writing" property when you use View Properties in the file system editor, or Show File Properties (m-X) in Zmacs. Files left in this state by crashes have to be removed manually by deleting and expunging. For example, suppose versions 3, 4, and 5 exist, but 5 is open in this state. An attempt to read **:newest** would get version 4; an attempt to write **:newest** would create version 6.

LMFS Multiple Partitions

The Lisp Machine File System (LMFS) utilizes one or more files of the FEP file system in which to store its files and directories. These FEP files are called *partitions*.

Normally, there is one partition, usually called LMFS.file.1. Alternatively, you can create multiple partitions on a disk. Although LMFS files reside inside these partitions, the FEP file system does not know about the files. It is the LMFS's job to manage LMFS files.

If you have multiple disks, and you want to use more than one of them on which to store LMFS files, then you must create at least one partition on each disk drive (since FEP file systems cannot span multiple disk drives). Then, LMFS must be instructed to use these partitions.

The selection of partitions to be used by LMFS is determined by a database called the File System Partition Table (FSPT). It is contained in a FEP file named FEP:>fspt.fspt.

The FSPT is optional. If it is not present, LMFS uses lmfs.file on the FEP boot drive. The FSPT is a simple character database containing the actual pathnames (in the FEP file system) of the partitions to be used for file system access.

If your machine has more than one disk, it may be difficult to find the disk location of the FSPT. In order to make finding the location of a FSPT easy, insert the Set LMFS FSPT Unit FEP command in your hello.boot file. This command causes LMFS to look for the file named FEP*n*:>fspt.fspt on the disk unit specified by *n*. For example, if you put your FSPT on disk unit 2, put the following in your hello.boot file:

```
Set LMFS FSPT Unit 2
```

Each partition in the file system knows how many partitions make up the file system. Only the FSPT, which is used only at LMFS startup time, indicates the locations of these partitions. That is, the file system databases in the actual partitions do not contain drive and partition numbers or FEP pathnames. Thus, before accessing the LMFS, partitions can be moved around using Copy File (m-X); as long as the FSPT is edited to indicate their new locations, LMFS comes up (when required) using the moved partitions.

Note: Since the Copy File (m-X) command copies files according to byte size, you may need to edit the byte count of the partition for the copy file command to work. To do this, multiply the number of blocks by **si:disk-block-length-in-bytes**, since partitions were previously created with a byte size of 0.

Edit the FSPT manually only to move partitions. When you add partitions to the file system, the File System Editor (FSEdit) automatically rewrites the FSPT database to include the locations of new partitions. See the section "Adding a LMFS Partition".

Adding a LMFS Partition

Partitions can be added to LMFS by following these steps:

1. Press SELECT F to select the File System Maintenance Program.

2. Click on [Local LMFS Operations] to invoke the second level of the File System Maintenance Program.

3. Click on [LMFS Maintenance Operations] to invoke the third level of the File System Maintenance Program.

4. Click Right on [Initialize] to invoke a menu of initialization options, which offers [New File System] and [Auxiliary Partition] as choices. Clicking on [New File System] is similar to clicking Left on [Initialize]; it initializes a partition to be the basis of a file system.

   Click on [Auxiliary Partition] to add an auxiliary partition.

   Enter the pathname of the FEP file to be used as the new partition. The default presented, which is correct for

   `[New File System]`

   , is never correct for adding an auxiliary partition.

5. Click on

   `[Do It]`

   . The system prompts for the number of blocks to allocate for the partition, and then performs verification and error checking. It must not be interrupted while performing these actions.

Free Records and the Free Record Map

The basic unit of allocation in the Lisp Machine File System is the *record*. A record is

`4 *` **si:disk-sector-data-size32**

(or four disk blocks). Each file system object is made from an integral number of records. At any time, each record is *in use* (representing an existing file system object) or *free* (not representing anything and free to be used in new objects).

When the file system needs to find a new free block to create or grow an object, it does not search through the records looking for a free one, because that would require many disk operations (and be slow). Instead, LMFS uses a redundant data structure called the *free record map*, kept in several blocks in a known location in the file system partition. The free record map uses one bit for each record in the file system; this bit marks whether the record is free or in use. The file system can find a free record quickly by examining this map.

If the file system crashes, or something else goes wrong, the contents of the free record map can become inconsistent with the contents of the file system itself. For each record, two different errors are possible (although the software is designed to prevent them):

1. The record might actually be in use, representing part of an object, but marked as "free" in the map.

2. The record might actually be free, but marked as "in use" in the map.

The first error is more serious than the second; the file system might use the record for a new object even though it is currently representing some existing ob-

ject, which could destroy the existing object. If the second error occurs, the record simply is not allocated. Such a record is said to be *lost*.

The file system is written so that a crash can only cause the second kind of error. While the file system is operating, it maintains a *free buffer* in its data structures in virtual memory. The free buffer is a pool of records that are not actually in use, but are marked as being in use in the free record map. When the system needs to allocate a record, it draws on one of these; when the system frees up a record, it adds it to this buffer.

When the free buffer gets too big, some records are removed from it and marked as "free" in the map on the disk; when the free buffer runs low, more records are marked as "in use" in the map, and are added to the buffer. If the machine is cold booted, or if the file system crashes, the records in the free buffer are lost (but no errors of the first kind are caused).

The size of the free buffer is maintained at about thirty records (so a crash loses about thirty records). To recover, log out of the machine or use the [Flush Free Buffer] command to flush the entire free buffer and mark the records as "free" in the map on the disk. To use the [Flush Free Buffer] command, press SELECT F to enter the File System Maintenance Program. Click right on [Local LMFS Operations] to invoke the second level of the progam, where you can click on [Flush Free Buffer].

After the buffer has been flushed, you can cold boot the machine without losing any blocks. Lost records can be found again by the Salvager. See the section "Running the LMFS Salvager".

You can check the number of free records in the file system by using the File System Maintenance program. First, press SELECT F to select the program. Then, click Right on [Local LMFS Operations], to invoke the second level of operations. In the second level, if you click Left on [Free Records], the program displays a line for each block of the file map, telling you which records are covered by that block, the number of such records, and how many are marked as free. It also tells you how many free records (marked as "in use" in the map) are in the free buffer, and finally displays the number of free, used, and total records in the file system.

To find out how many records are actually in use, click Middle on [Free Records] to prepare a printable report of record use throughout the file system. This has to pass over every object in the file system, and so it takes some time, especially on large file systems. The discrepancy between the answer of this function and the answer you get when you click Left on [Free Records], tells you how many lost records there are; if there are a lot, you might want to run the Salvager.

Clicking Right on [Free Records] displays how many records are in use in each partition. This information is necessary for commands such as [Grow Partition] that allow you to change the size of partitions, add partitions, or remove partitions.

Running the LMFS Salvager

The *Salvager* is a program that reads every LMFS record of the file system and finds and fixes certain inconsistencies and errors. The Salvager can:

- See which records are in use and which are free, and update the free record map to reflect the current state of the file system. (This is how you recover lost records.)

- Find objects that are stored in a file system partition but are not referenced by any directory. Such objects are called *orphans*; they exist only if some problem has occurred, such as a file system crash during the creation of a file. When the Salvager finds such objects, it puts them back into the directory hierarchy (*repatriates* them).

The Salvager always reconstructs the free record maps. Running the Salvager takes about two minutes per thousand records of file partition.

When the Salvager is repatriating an orphan and it cannot find the directory in which the orphan is supposed to reside, it creates a new directory as an inferior of the directory >repatriations, with a name like lost-1 or lost-2. After a Salvager run, you should examine these directories.

The Salvager always considers storage occupied by orphans to be "in use" for purposes of the free record map. If many orphans existed, they would use up a great deal of disk space. (Normally, however, orphans do not exist.)

When the Salvager repatriates, it also "fixes" disk errors and misplaced records or directories by replacing them with fresh, empty ones. By nature of the process, no files are lost during repatriation.

Note: When the Salvager repatriates an object, it types out a message saying that it did so. One of these messages might cause a **MORE** pause. If you plan to leave your console unattended while the Salvager is running, you might want to disable **MORE** pauses before you leave.

To run the LMFS Salvager, press SELECT F to select the File System Maintenance program. Click on [Local LMFS Operations] to invoke the second level of the program. Next, click on [LMFS Maintenance Operations] to invoke the third level of the program. Then click on [Salvage] to obtain a menu of options.

If you have a local file system of multiple partitions (occupying multiple FEP files), you are presented with a menu of partitions to process. This menu, which is an Accept-Values menu, also includes questions about Salvager operations. In addition to listing the partitions to be salvaged, the menu offers you the options as shown in Figure !.

Here are the options:

```
Top-down treewalk record check: yes no

Check for and repatriate orphans: yes no

Output recording: Tape File Console only

File for output:
```

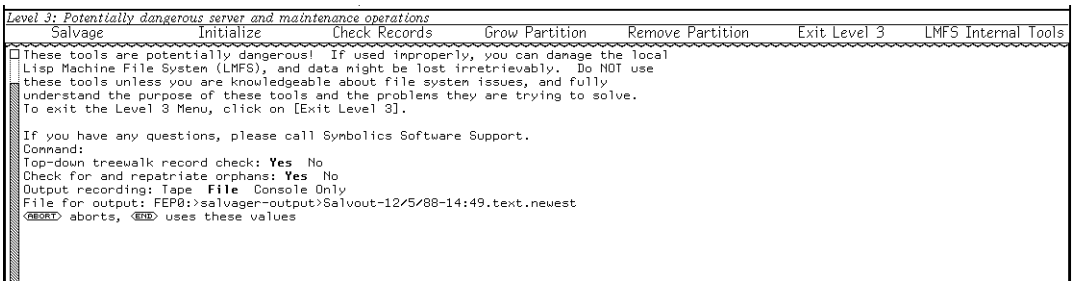


Figure 107. Salvager Options

The first items on the menu constitute a list of partitions you can select for processing by the Salvager. You can choose some or all of the partitions for processing.

The second menu option, `Top-down treewalk record check`, offers to scan all of the directories and files in the local LMFS and report any damaged records (hardware or software), disappeared files, or any other problems. This search starts at the root and goes through all of the file system, directory by directory, and is performed after all other salvaging activity.

Note: If you deselect any partition for repatriation, then the next menu item, which offers to check for and repatriate orphans, disappears. This happens because it is impossible to construct an accurate model of the hierarchy if each partition is not scanned.

The third menu option, `Check for and repatriate orphans`, offers to find orphaned objects and put them back into the directory hierarchy. During this scan, the Salvager also replaces bad directory records with good ones.

The fourth menu option, `Output recordings`, offers to log the Salvager output either to tape, in a file, or only to the console.

- If you choose the `Tape` option for output recording, every message goes onto the tape as soon as it is produced because of a special format that is used. Using an industry-compatible tape ensures that all messages appear on tape. If you use a cartridge tape, this is not fully guaranteed.

  The following forms can be used to view the tape produced in this way:
  lmfs:print-salvager-output-tape

  lmfs:copy-salvager-output-tape-to-file

- If you choose the `File` option for output recording, you must supply a filename. The filename can be a pathname on the local FEPFS, or a pathname on the FEPFS or LMFS of another host. The filename cannot be a pathname on the local machine's LMFS. The default file is a FEP file, on boot unit 0. Every time a

line of Salvager output is written to a file, a **:Finish** is done to the file, so that even if the system crashes, the file is intact with all the Salvager output up to the point of failure.

If you decide to put the output recording in a FEP file, make sure there is enough room, probably about 100 blocks. If you have your output recording sent to another host, choose a host that you are sure will stay on the network during the logging process.

There are currently no tools for automatically processing a file containing a log of Salvager output.

- If you choose the `Console only` option for output recording, note that this is not usually the device of choice. You should choose this option when there is no other means of logging available.

- If any problems occur while the log is running, such as a file closing or a disrupted network connection, a menu appears. This menu asks what to do about continuing the Salvager's log. If you enter the Debugger while the log is being recorded, you are offered restart options for discontinuing or reselecting log options.

Salvager-Related Lisp Functions

The **lmfs:print-salvager-output-tape** and **lmfs:copy-salvager-output-tape-to-file** Lisp functions are useful when running the Salvager.

lmfs:print-salvager-output-tape &optional *tape-spec* (*stream* **zl:standard-output**)
Function

Prints the contents of the tape created by the Salvager. If you do not supply any arguments, you are prompted for a tape spec, and output prints to the console.

lmfs:copy-salvager-output-tape-to-file &optional *tape-spec pathname* *Function*

Prints the contents of the tape created by the Salvager to a file. You are prompted for any arguments not given.

LMFS Links

A link is a file system object that points to some other file system object. Thus, if you want a file called >George>Sample.lisp appear in the >Fred directory, with the name New.lisp, you can create a link by that name to the file. Then if you open >Fred>New.lisp, you really get >George>Sample.lisp. The object to which a link points is called the *target* of the link, and can be found from the **:link-to** property of the link.

The above explanation is simplified. You might wonder if, for example, you try to rename >Fred>New.lisp, is the link — or the target — renamed? Each link has a property called its **:link-transparencies**. The value of this property is a list of keyword symbols. Each symbol specifies an operation to which the link is transparent. If the link is transparent to an operation, then when the operation is performed, it is performed on the target. If the link is not transparent to the operation, the operation is performed on the link. Here is a list of the keywords, and the operations to which they refer:

:read	Opening the file for **:input**.
:write	Opening the file for appending, via **:if-exists :append**.
:create	Opening the file for **:output**
:rename	Renaming the file.
:delete	Deleting the file.

You can create new links with the [Create Link] command in the File System Editor, or Create Link (m-X) in Zmacs.

Programs can use the **:create-link** message to pathnames. See the section "Pathname Messages: Naming of Files".

When a new link is created, its transparencies are set from the **:default-link-transparencies** property of its superior directory. When a new directory is created, its **:default-link-transparencies** property is set to **(:read :write)**.

The value of the **:link-transparencies** property is a list of keywords describing the transparency attributes of a link. The value of the **:default-link-transparencies** attribute of a directory is, similarly, a list of all those transparencies for the newly created links in this directory. When changing the value of either of these properties with **fs:change-file-properties**, the new value of the property is a list of transparency keywords. Transparencies not present in the new value are turned off, and they are not preserved. There is no way to change an individual transparency.

When you create a new link with the [Create Link] command, specify both the name and the type component of the new link; the version defaults to newest, as soon as you create the link.

When you specify the target, give a complete pathname consisting of name and type; the version can be unspecified. Whenever the targets of links have unspecified versions, the versions are treated as **:newest**.

There is a subtle point regarding "create-through" links (links transparent to **:create**). If a pathname is opened for **:output** (which means it is being created), and the pathname has version **:newest** or a version number that is, in fact, the newest one, and the newest version is actually a create-through link, the link is transparent and the operation is performed in the target's directory.

If the target pathname has a version, it is as if that exact pathname were opened for **:output**; if the target has no version, it is as if the target pathname with a version of **:newest** were opened.

A directory link is a link whose type is "directory", whose version is 1, and whose target is a real directory or another directory link. The maximum permitted length of such directory link chains is 10. The system respects a directory link when looking for a directory. By means of directory links, "indirect pointers" (or multiple names for directories) can be established. No special action needs to be taken in order to declare a link to be a directory link. Transparencies are not interpreted in directory links.

Dumping, Reloading, and Retrieving LMFS Files

A file system can be damaged or destroyed in any number of ways. Users can delete files by accident. To guard against such a disaster, it is wise to *dump* the file system periodically, that is, write out the contents of the files, their properties, and the directory information onto magnetic tapes. If the file system is destroyed, it can then be *reloaded* from the tapes. Individual files can also be *retrieved* from tapes, in case a single file is destroyed, or just accidentally deleted (and expunged). Dump tapes can also be used to save a copy of all the files on a machine for archival storage.

In a *complete dump*, all of the files, directories, and links in the file system are written out to tapes. This, obviously, saves all the information needed to reload the file system. However, a complete dump can take a long time and use a lot of tape, especially if the file system is large. In order to make it practical and convenient to dump the file system at short intervals, a second kind of dump can be done, called an *incremental dump*.

In an incremental dump, only those files and links that have been created or modified since the last dump (of either kind) are dumped; things that have stayed the same are not dumped. (All directories are always dumped in an incremental dump.) Now, if the file system is destroyed, you reload it by first reloading from the most recent complete dump and then reloading each of the incremental dump tapes made since that complete dump, in the same order in which they were created. Therefore, you do not need to retain incremental dump tapes that were made *before* the most recent complete dump was done; you can reuse those tapes for future dumps.

Since all tapes containing incremental dumps done since the last successful complete dump must be reloaded in order to restore the file system, doing a complete dump regularly makes recovery time faster. Doing complete dumps also lets you reuse incremental dump tapes, as described above. The more incremental dump tapes you must load at recovery time, the longer it takes to recover, and thus the more chance there is that something will go wrong. Thus, it is advantageous to perform complete dumps periodically.

A *consolidated dump* is like an incremental dump, in that it only dumps files that have been created or changed recently. However, a consolidated dump backs up only those files that have been created or changed since a specified *consolidation date*. A consolidated dump is the appropriate kind to take if some event destroys recent incremental dump tapes, or they are found to be unreadable. If a complete dump extends through several days, it is wise to take an incremental dump between tape stopping points as appropriate.

Dumping LMFS Files to Tape

To dump LMFS files to tape, follow these steps:

1. Mount a magnetic tape on a tape drive that's connected to the local machine, or connected to a machine that the local machine can access over the network.

2. Press SELECT F to select the File System Maintenance Program and click on [Local LMFS Operations].

   This invokes the second level of the File System Maintenance Program, which is called *Local File System Control Operations*.

3. Choose either [Incremental Dump], [Complete Dump], or [Consolidated Dump] by clicking on the menu.

   These commands respond with an Accept-Values menu that lets you set the parameters of the dump; the only difference among the commands is the initial value of some of the parameters in this window.

4. Change the values in this window as needed.

Figure ! shows the second level of the File System Maintenance Program.

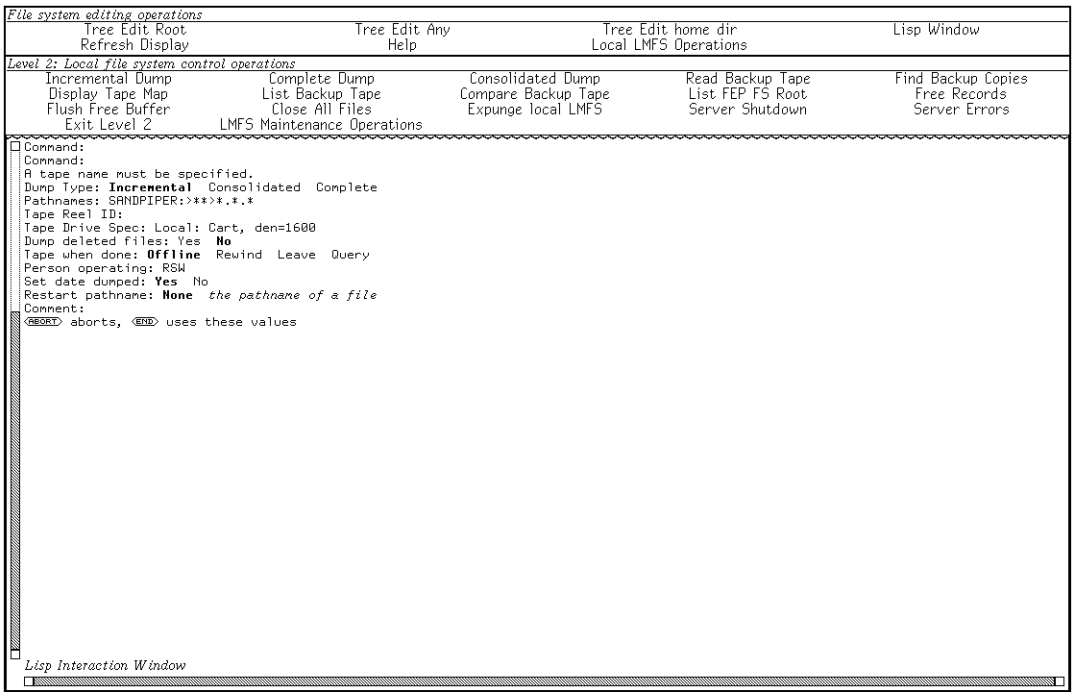


Figure 108. Dumping LMFS Files To Tape

Here is an explanation of the parameters offered for modification in this window:

Dump Type

There are three possible types of dump: *incremental*, *consolidated*, and *complete*. The three File System Maintenance commands initialize this field according to their own requirements; the [Dump] command initializes it to *complete*.

Pathnames

The pathname, or pathnames, specifying what is to be dumped. If there is more than one pathname, they are separated by commas. This value controls what files and directories are inspected for dumping. The type of the dump (complete, incremental, or consolidated) and the status of the individual files controls what subset of these files are actually dumped. For information about the different types of dumps: See the section "Dumping, Reloading, and Retrieving LMFS Files". Names of single files or links can be used to dump single files or links.

Wildcard specifications can also be used: this is the normal way to dump many files from one directory, or from a subtree. Subtrees are dumped via recursive (**:wild-inferiors**, "**") directory wildcards. The pathname you type is merged with "local:>**>*.*.*".

To dump the whole file system, which is the normal default, the appropriate pathname is:

```
>**>*.*.*
```

To dump all the files in directory >foo>bar, and all of its inferiors, the appropriate pathname is:

```
>foo>bar>**>*.*.*
```

To dump all the latest Lisp files in directory >abel>baker, but not any of its inferiors, the appropriate pathname is:

```
>abel>baker>*.lisp.newest
```

See the section "Naming of Files". See the section "LMFS Pathnames".

Tape Reel ID

Every reel of tape produced by the dumper must have a Tape Reel ID, which is a string of up to eight characters. You must explicitly supply a value for this option. The reel ID is used to identify this reel of tape to the backup system; it appears in the dump maps and in any messages about the tape. The **:complete-dump-tape** or **:incremental-dump-tape** property of any file dumped is set to this value, as well. The Tape Reel ID should be written with a pen onto the label of the tape so that the tape can be identified by sight. You must supply a Tape Reel ID.

Tape Drive Spec — A tape specification that describes the host and the tape drive to be used. The default is usually **local: cart** (if there is a cartridge tape drive present on the local host). For more information, see the section "Tape Specifications".

Dump Deleted Files — Yes or No; this says whether files marked as deleted but not yet expunged should be dumped onto the backup tape. The default value is No; deleted files are normally not dumped.

Tape when done — This controls what the dumper does with the tape when it has finished passing over all the specified pathnames. These are the available options:

- Offline — Rewinds the tape and puts it offline. It declares the dump finished. This is the default.
- Rewind — Rewinds the tape without putting it offline. It declares the dump finished. It facilitates listing or verifying the tape contents.
- Leave — Leaves the tape positioned at the end without rewinding it or putting it offline. It declares the dump finished. It facilitates more dumping later, by leaving the tape in the correct position for using "Append to tape" in a later dump invocation.
- Query — At the end of the dump, all of these options are presented, and you can choose whether to rewind and set the tape offline, rewind it, leave it at end, or dump some more files. If you choose to dump more files, the dumper menu is offered again, and the new files are appended to this tape. The dump is not declared finished until you click on "Abort".

Person operating — The identification of the person doing the dump. This is entered into the backup map, and sets this person as the file author of that map. Normally, this is the same as the login ID of the user performing the dump, and that is its default value. However, if the user who is performing the dump is not logged in to the machine from which the dump is invoked, this field should be filled with that user's name. It is important for documentation purposes and site record-keeping.

Consolidate from — This field is only used during a consolidated dump. It is a date and time in the past, entered in any acceptable Genera

format. The consolidated dump dumps all, and only, files that have been created or modified since this date.

Set date dumped — This can be either Yes or No. The default is Yes, so that when the dumper finishes writing a tape, it marks all the files it has dumped as having been dumped on that tape at this time, and creates a *tape directory*, as described below. These measures allow the file to be retrieved later, and indicate that the file no longer needs to be dumped in incremental dumps. This is the default action, which corresponds to a value of Yes. **Note:** The LMFS dumper should not be used to move software between sites as it is far too general, and system-independent; use the carry system and the distribution tape system instead. However, if you do use the dumper to make tapes that are not part of your site's backup, such as for moving software between machines, you do *not* want to indicate that the files were dumped, or to make a tape directory. Select No in this case.

Restart pathname — The purpose of this feature is to allow restarting of complete dumps that are interrupted by any sort of failure. When the dumper finishes a tape, it prints out the pathname of the last file dumped. Although this is recorded in the dump map, *the pathname and the name of the tape should be recorded on paper by the person doing the dump*, especially if a complete dump is being done.

To restart a dump, fill out the menu as usual, but type in the pathname of the last file known to have been dumped as the value of [Restart Pathname]. The dumper scans the subhierarchy indicated, but does not dump files already dumped. The dumper does not scan directories that the restart pathname indicates have already been processed. The skipping of files and directories already dumped is based on sorting order, not whether the file has actually been dumped. Thus, if files A, C, E, G, and I exist, and files A through E get dumped one day, and the dump is interrupted and restarted from E the next day, a D created in the interim is not dumped.

Comment — A string, of arbitrary contents, written on each reel of the dump and in the dump map. This might say why the dump was performed, or any other special information about this dump.

When you are done filling in values, press END; if you decide not to do a dump after all, press ABORT. If there is something wrong about the set of parameters you have specified, the program displays a message and presents you with the Accept Values window again. Otherwise, it displays a message saying that the dump has started successfully, and proceeds. While the dump is in progress, the name of the

file that is being dumped is shown in the far right-hand field of the status line at the bottom of your screen (this is the field that normally shows you the names of files that are being read or written).

The dumper creates a file called the *dump map*. The dump map is a character file, giving a complete description of what has been dumped, directory by directory and file by file, including the time of dumping, the tape on which the file was dumped, the tape reel ID of the previous tape on which the file appears (if any), and so on. The dump map is created in the >dump-maps directory. Its name is constructed from the type of dump and the date and time at which it was started; the file type is **map** and the version is **1**. A typical dump map might have the pathname:

```
>dump-maps>complete-3/15/88-9:02.map.1
```

The dumper puts all information about the dump, the operator, the time of day, the options, and so forth, in the map. It also puts error recovery information there, and descriptions of tape-changings, as well as the number of files dumped on each tape. The dumper performs a **:finish** operation on the map file at the end of each tape, so that if the system crashes during a multi-tape dump, information about previous tapes is guaranteed to be intact and accessible.

The dumper also creates a file called the *tape directory* for each separate reel of each dump. This is a binary file saying what is on the tape, with more or less the same information as the dump map. You use this file when you try to locate dumped copies of a file. See the section "Finding Backup Copies of LMFS Files". The tape directory is also created in the >dump-maps directory. Its name is the tape reel ID of the tape, its file type is **directory**, and its version is **1**. A typical directory map might have the pathname:

```
>dump-maps>INC00001.directory.1
```

The dumper dumps files successively to tape, and at the end of each tape, rewinds and unloads the tape, asking for a new tape if there are more files to be dumped. It is only after it has done this that it sets backup dates for the files and makes the dump directory.

If the dumper gets an irrecoverable error while writing a tape, it attempts to write end-of-file marks on the tape and inform you of what has happened. It gives you the option to either keep considering the files on that tape to have been validly dumped, in which case the dump continues on the new tape, or discarding the discard tape, in which case it redumps all the files that it had dumped on the bad tape onto the new tapes. The problem and its chosen recovery are described in the dump map.

By default, the dumper tries to read each tape before writing on it. This is to avoid accidentally overwriting valuable tapes. For tapes to be appended to, this is necessary. For other tapes, it is desirable. It often takes a long time to attempt to read blank tape, to prove that a new tape is really new. The dumper explains and queries if it is not confident that the tape being written on is the right one.

Some sites may want to waive this checking. This is necessary when tape hardware is in use that cannot time out while reading blank tape, and therefore reads the whole tape when a new tape is checked, with no way to stop it. This checking

is controlled by the site option **validate-lmfs-dump-tapes**, an attribute of the Site object for a site. The value of this attribute is normally Yes, and it enables suppression of this checking. To supress tape checking, change the value of this attribute to No.

You can verify that a newly made dump tape contains good data by using the reloader's [Compare] option in the Read Backup Tape menu. See the section "Reloading and Retrieving LMFS Files". For information on comparing backup tapes: See the section "Comparing Backup Tapes".

Finding Backup Copies of LMFS Files

In order to retrieve individual files, or groups of files, rather than reloading an entire tape, you must know on what tape the files were dumped. The LMFS backup mechanism provides a way to search the binary tape directories, which are produced by the dumper, to find backup copies of files. This is the Lisp function **lmfs:find-backup-copies**, which you can invoke by clicking on [Local LMFS Operations] in Level 1 of the Files System Editing Operations program. Then click on [Find Backup Copies]. This prompts you for pathnames.

lmfs:find-backup-copies searches all the binary tape maps at the site for the tape locations of all backup copies of a file. It prompts for names, which it parses with respect to the local host unless you name an explicit host in the pathname. All the files requested must be from the same host. This applies only to Symbolics hosts. It looks at the binary tape maps on the specified host in the directory >dump-maps.

Normally, you specify wildcard pathnames for this function to match. A non-wild pathname is considered to be a valid degenerate case of a wildcard pathname. The default with which all of the names are merged is "local:>**>*.*.*". This function uses the same pathname interpretation conventions as the reloader.

Here is a sample interaction with **lmfs:find-backup-copies**:

```
Enter file pathnames for which to search, separated by commas.
Wildcard names are allowed.  [default LOCAL-LISPM:>**>*.*.*:]
Paths: f:>sys>io>*.lisp, f:>bsg>*.init, f:>lisp>*.q*
Searching for:
  F:>sys>io>*.lisp.*
  F:>bsg>*.init.*
  F:>lisp*>**>*.q*.*
-----
F:>LISPM>COLDLD.QBIN.39, created 2/17/88 00:26:58, on tape fscns001
  (Backup dump of 2/24/88 13:05:44)
F:>LISPM>COLDUT.QFASL.57, created 2/18/88 19:06:36, on tape fscns001
  (Backup dump of 2/24/88 13:05:44)
F:>LISPM>UTILS>NEW>EVAL.QBIN.13, created 2/19/88 04:45:17, on tape fscns001
  (Backup dump of 2/24/88 13:05:44)
```

....and so on
----- Scan complete.

Reloading and Retrieving LMFS Files

Reloading is the process of moving all the files on a backup tape into the LMFS of a target host, which can be local or remote. *Retrieving* is the process of moving only selected files from tape into the LMFS.

Reloading and retrieving can load files into any LMFS.

Two other functions are related to reloading and retrieving: listing the contents of backup tapes with the List tape option, and verifying the contents of the dump with the Compare option. The *reloader* program implements all four of these functions.

To invoke the reloader:

1. Press SELECT F to get to Level 1 of the File System Maintenance Program.

2. Click on [Local LMFS Operations] to invoke Level 2.

3. Click on [Read Backup Tape].

Figure ! shows the Accept-Values menu that appears when you click on [Read Backup Tape].

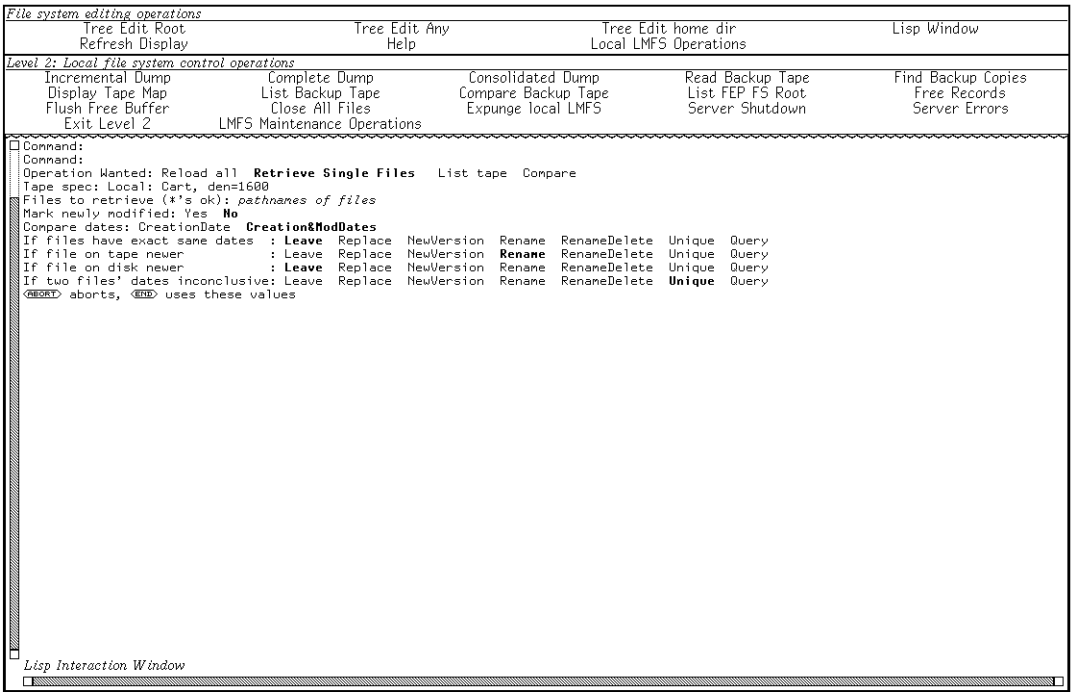


Figure 109. The Read Backup Tape Menu

The reloader Accept-Values menu contains the following items:

Operation Wanted This selects which function the reloader is to perform.

These are your choices. To specify a remote host, precede the pathname with *remote-host-name:>* .

Reload all
: The reloader is to read the tape, and reload all files on it. It will not reload files that are already present on the target LMFS.

Retrieve Single Files
: The reloader will expect a list of pathnames, to tell it which files to reload. It will not reload files that are already present on the target LMFS. You specify these pathnames by clicking on the menu item [Files to retrieve]. You can specify wildcard pathnames. For example, a wildcarded pathname for a single file on a remote host looks like this:

```
E:>trees>.*
```

List tape
: The reloader will read the tape and display a description of its contents on the screen. It lists the dumps appearing on the tape, and which files are on the tape. This does *not* verify that the files were dumped correctly; use "Compare" for that.

Compare
: The reloader will read each file on the tape, look for the same file in the file system, and compare the two, bit for bit, reporting any discrepancies. Use this option to verify that a dump tape just created contains good data.

After selecting the operation to be performed, supply the following information:

Tape Spec:
: A tape specification that describes the host and the tape drive to be used. The default is usually **local: cart** (if there is a cartridge tape drive present on the local host). For more information, see the section "Tape Specifications".

Files to retrieve (*'s ok):
: One or more pathnames, which are usually wildcard pathnames. This is used when you click on [Retrieve single files]. Any file matching one of these wildcard pathnames will be reloaded, unless it is already there. For more information:

  - See the section "Naming of Files".
  - See the section "Wildcard Pathname Mapping".
  - See the section "LMFS Pathnames".

Mark newly modified: Yes No
: Specify yes to mark the newly reloaded files as *not yet dumped*

so that they will get dumped by the next incremental or consolidated dump. Choose no if you don't want the files to be dumped in the next dump. The default is No.

The reloader will move all the files from the backup tape into the target file system unless files of the same name, type, and version exist. If this is the case, the reloader compares the creation and or creation and modification dates of these files according to guidelines you specify here. These are the guidelines you can choose:

1. Compare Dates: CreationDate Creation&ModDates. This compares the creation dates or creation dates and modification dates of the files you are trying to reload, depending on what you choose.

   If you choose Creation&ModDates, this option is presented: Leave, Replace, NewVersion, Rename, Unique, Query. The default is Unique. The meaning of these choices is explained below.

   a. If files have the exact same dates, your choices are: Leave, Replace, NewVersion, Rename, RenameDelete, Unique, Query. The default is Leave. The meaning of these choices is explained below.

   b. If files on tape are newer, your choices are: Leave, Replace, NewVersion, Rename, Unique, Query. The default is Rename. The meaning of these choices is explained below.

   c. If files on disk are newer, your choices are: Leave, Replace, NewVersion, Rename, Unique, Query. The default is Leave. The meaning of these choices is explained below.

Here is the meaning of each choice:

Leave	Leaves the file in the file system and does not reload it.
Replace	Loads the file from tape, completely replacing the file in the file system.
NewVersion	Loads the file from tape as the newest version.
Rename	Renames the file uniquely, and loads the file from tape.
RenameDelete	Renames the file uniquely, deletes it, and loads the file from tape.
Query	Stops the reload and asks you what to do when the tape is ready.
Unique	Loads the file from tape under a unique name in the proper directory.

When you have made your choices from the reloader Accept-Values window, press END to begin the reloader. If you do not wish to proceed with the reload, tape list, or compare, press ABORT.

The reloader prints information about each file it reloads or about every file on the tape, if you have selected [List tape]. When it gets to the end of the tape, it stops. If you want to continue reloading, you must mount another tape and restart the reloader. Tapes can be reloaded in any order, but choose your reload options carefully. Generally you should load tapes in the opposite order of creation, and not replace any newer files. This prevents writing over files, such as mailboxes.

The reloader does not delete or expunge files. If you are reconstructing a file system from backup tapes that include many incremental backups, you must occasionally intervene to delete and expunge unwanted old files that are reloaded, in order to ensure adequate file space.

Comparing Backup Tapes

A few notes about the comparer: Occasionally, hardware problems can cause bad backup tapes to be written, without any error having been detected by the dumper. You should always verify a backup by using the "Compare" option of the reloader to *verify* the backup tape. The comparer verifies each bit of every file on the tape, and, for files that still exist, reports any discrepancy.

If you find that an incremental backup tape is deficient, and you decide that the files must be redumped, you must perform a consolidated dump, with a consolidation date equal to the time the bad incremental dump started. You must delete the backup dump tape directory ("*tape-name*.directory" in >dump-maps) by hand, if the backup-copy finding mechanism is to be aware that the tape has been abandoned.

The reloader produces an error log file in the target directory >reloader-logs.

The dumper assumes that no undetected problems occurred. Thus, it does not run the comparer automatically. The dumper sets backup times when it finishes each tape.

File System Maintenance Program

The File System Maintenance Program includes the File System Editor (which is the only part that most File System Maintenance Program users ever use).

Besides the File System Editor, the File System Maintenance Program provides tools for doing file system maintenance operations, such as manipulating backup tapes, and running the Salvager.

Although it includes commands for manipulating remote file systems, including the interface to the File System Editor, the File System Maintenance Program is intended primarily for manipulating the local file system.

To enter the File System Maintenance Program:

- Press `SELECT F`.
- Issue the Command Processor (CP) Select Activity File System Operations command at the Command Processor prompt.

- Click on File System in the System menu.

The File System Editor (FSEdit) is an interactive program that lets you examine and modify the contents of a file system. Using it, you can create directories and links, view and edit the properties of file system objects, delete objects, and expunge directories.

Figure ! shows the initial window of the File System Maintenance Program.

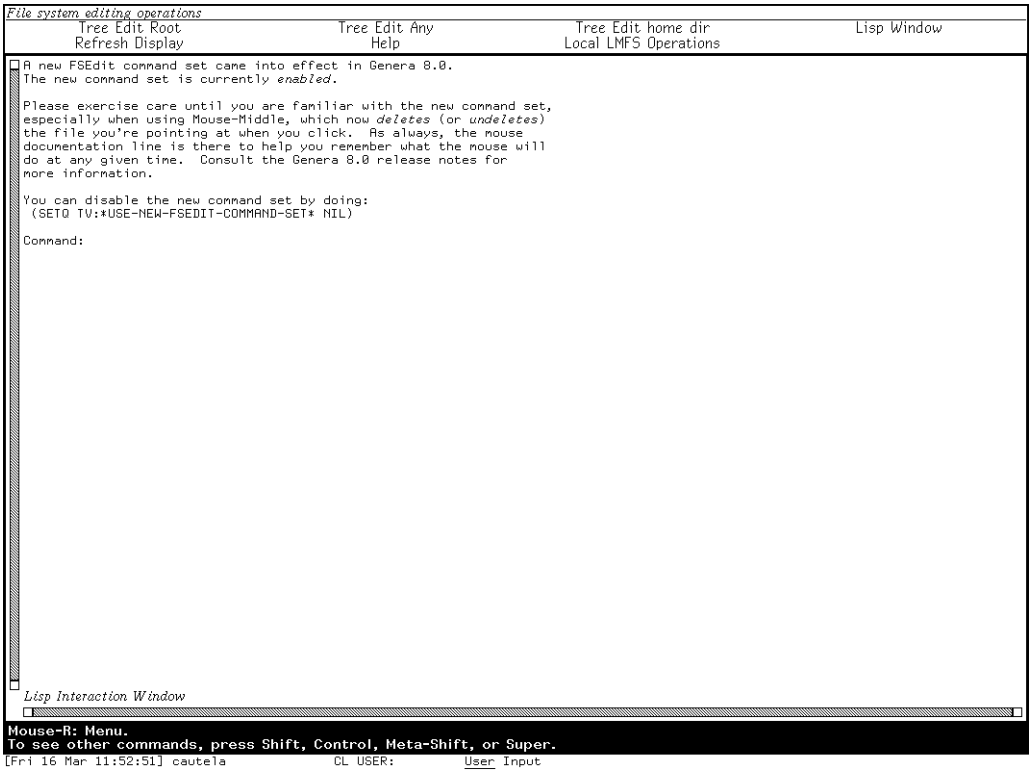


Figure 110. Initial File System Maintenance Operations Menu

The File System Maintenance Program uses a frame with a command menu pane at the top and a large pane beneath it. You give commands by clicking on the command menu pane. The large pane is initially a *Lisp Interaction Window*. The menu has four levels of operations, which unfold as you click on certain operations. For example, clicking on [Local LMFS Operations] in the first level of the command menu invokes the second level. Clicking on [LMFS Maintenance Operations] invokes the third level. Then, clicking on [LMFS Internal Tools] invokes the fourth level of operations. Figure! shows the four menu levels.

In addition, if you are in the first level of the menu, which is initially a *Lisp Interaction Window*, and you click on a File System Editor command in the command menu (for example, [Tree Edit Root]), the large window changes to a *File System Editor* window. If you click on [Lisp Window], this command changes the large window back to a Lisp Interaction Window.

When the large window is a Lisp Listener, you can type Lisp forms there and have them evaluated and printed. When it is a File System Editor window, you are in the File System Editor and can click on the items in the menu to activate File System Editor commands.

Using FSEdit

When you use [Tree Edit Root], at the top of the main window is a line reading `>*.*.*`. This line represents the root directory, which usually contains only directories. Below the root directory line is a set of indented lines, one representing each object in the root directory.

Move the mouse over any one of these directory lines and notice that the mouse documentation line reflects three actions that you can take:

(L)	Open/Close object. If the object is a directory, it shows or hides the directory contents. If the object is a file, it does Show File on the file.
(M)	Delete/Undelete object.
(R)	Menu of operations. See the section "FSEdit Commands".

By default, deletion via this command always requires confirmation. You can set the variable **tv:*confirm-fsedit-quick-soft-file-deletion*** to **nil** if you want to disable the confirmation of *soft* deletion (deletion of a file on a host which supports undeletion). No mechanism is provided for disabling confirmation on a host that does not have soft deletion.

Interpreting Directory Listings in FSEdit

The system displays the contents of directories of file systems in three contexts:

- The File System Editor
- The Show Directory (`m-X`) and Dired (`m-X`) Zmacs commands
- The Show Directory command

Contents of directories are displayed in a standard format, regardless of the context and regardless of what kind of file system (for example, Symbolics computer, TOPS-20, UNIX) from which the directory came.

This format is abbreviated because it is designed to express a great deal of information in a single line. The basic format looks similar to this:

```
pal.lisp.65    7  25548(8)    03/12/85 12:42:41 (05/13/85)   dlw
```

The following is an explanation of the items in this listing:

item	*explanation*
`pal`	file name
`lisp`	file type
`65`	file version number

7	length of the file in blocks
25548	length of the file in bytes
8	byte-size of the file
03/12/85	date file created
12:42:41	time file created
5/13/85	date file last referred to
dlw	author

Many other things can appear in such a line; some of these things are seen only on certain types of file systems. If the first character in the line is a D, the file has been deleted (this makes sense only on file systems that support undeletion, such as the Lisp Machine and TOPS-20 file systems). After the D, if any, and before the name of the file, is the name of the physical volume that the file is stored on (on ITS, this is the disk-pack number).

On a line that describes a link rather than a file, the length numbers are replaced by an arrow (=>), followed by the name of the target of the link.

On a line that describes a subdirectory rather than a file, the length-in-blocks number is shown (if provided by the file system), but the length-in-bytes is replaced by the string DIRECTORY.

Next, before the dates, the line might contain any of several punctuation characters indicating things about the file. Only some of the file systems understand these flags. Following is a list of the various characters and the flags they indicate:

character	*flag*
!	not backed up
@	do not delete
$	do not reap

For lines indicating subdirectories, the reference date can be replaced with a date preceded by X=, the date this directory was last expunged. The dates are followed by the file author's name, which is followed by the name of the last user to read the file.

Only certain file systems support certain features. Many file systems do not keep track of the last reader's name and do not have something comparable to a "do not delete" flag. Therefore, any of the above fields might be omitted on certain file systems. However, the same general format is followed for all file systems and so you can interpret the meaning of a line in a directory listing, even for a file system with which you are not familiar.

Opening and Closing a Directory in FSEdit

You can open and close a directory by moving the mouse over the line representing a directory and clicking Left. When you open a directory, a line is inserted in the display for each object in the directory. For every directory, there is a line with the pathname of the directory and nothing else; these directories are all

closed. For every file, there is a line with the name, type, and version of the file, and other information about the file. For every link, there is a line with the name, type, and version of the link, followed by => and the pathname of the target of the link, and other information. See the section "Interpreting Directory Listings in FSEdit".

Whenever you click Left on a closed directory, FSEdit opens it and displays its contents. By clicking on successive directories inside other directories, you can move around in the file system and see what is there. The base directory is automatically opened as soon as you start using the File System Editor.

When you are finished with a directory, you can *close* the directory by clicking Left on it again.

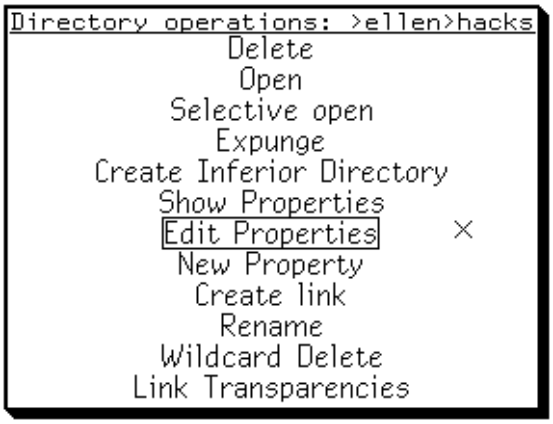
Using these commands, you can get at any part of the file system underneath the base directory, and see everything that is there.

It is easy for the display to become longer than the size of the window when you move around in large directories; you can use the usual mouse scrolling commands to move the display up and down in the window. See the section "Scrolling with the Mouse".

FSEdit Commands

To operate on an object, click Right on it. This pops up a menu of commands, each of which specifies an action to take regarding the object.

The menu for a directory looks like this:



Some commands make sense for all three kinds of objects (directories, files and links); others pertain only to certain kinds of objects. The menu that appears when you click right on an object offers only the options that you can apply to that type of object.

For example, the menu does not display [Expunge] as an option for files or links, only for directories, and it does not display [Expunge] an option if the directory in question resides on a host that does not support soft deletion.

The following is a list of FSEdit commands and the object(s) to which each command applies:

[Delete] *(Files, directories, links)*
Marks this object for deletion. This command pertains to systems that support soft deletion, for example, Genera. This command is only displayed for objects that are not already deleted. You should not delete directories that have anything in them.

[Delete (immediate)] *(Files, directories, links)*
Deletes this object. This command pertains to systems that do not support soft deletion, for example, UNIX. This command asks for confirmation and then immediately removes the deleted object from the display. You should not delete directories that have anything in them.

[Wildcard Delete] *(Directories)*
Does wildcard deletion. This command prompts you with a default for deleting everything for the line to which the menu applies. It merges what you enter with * defaults. It lists the files it intends to delete, asks for confirmation, deletes them, reporting any errors, and updates the display.

[Undelete] *(Files, directories, links)*
Undeletes this object. This command pertains to systems that support soft deletion, for example, Genera, and is displayed only for objects that are deleted (are marked with a D).

[Rename] *(Files, directories, links)*
Renames this object; prompts for a new name. If the object is not a directory, you can optionally type in a whole pathname specifying a new directory, and the file or link will be moved to the new directory as well as being given the new name.

[Show Properties] *(Files, directories, links)*
Types out one line for each property of the object, giving the name and the value of the property.

Properties are the qualities of the file that are maintained by the file system on which it resides, such as creation date and time, author, time of last access, and length. For files on a Symbolics File System, this means user-defined properties as well. It prompts for the name of a file and pops up a choose-variable-values window, allowing you to alter various properties of the file. The exact properties that can be altered depend on the file system, but they might include:

- Generation (version) retention count
- Author
- Creation, modification, and reference dates
- Protection flags
- Other file-associated information

This information types out on top of the display, and prompts you to type any character when you are ready to proceed. After you type this character, the properties vanish and the FSEdit window is redisplayed. You can also use [Refresh Dis-

play] in the command menu to make the typeout vanish; this is convenient since you do not have to move from the mouse to the keyboard.

[Edit Properties] *(Files, directories, links)*
Pops up a Choose Variable Values window that lets you change the value of any changeable system property or user property of the object.

[New Property] *(Files, directories, links)*
Creates a new user property for the object. You are first prompted for the name of the property, and then the value. The name is uppercased. To remove a property, give an empty string as the value.

[Show] *(Files, links)*
Displays the file. The file is typed out on top of the display, and you are prompted to type any character when you are ready to proceed. The **:reference-date** of the file of the file is not changed. See the section "LMFS Properties". If the object is a link, it must be transparent to **:read** and its target must be a file; the target is printed.

[Create Inferior Directory] *(Directories)*
Creates a new directory inside this directory. You are prompted for the name (just type in the name, not the whole pathname).

[Create Link] *(Directories)*
Creates a new link inside this directory. You are first prompted for the name of the link, and then for the full pathname of the target of the link. See the section "LMFS Links".

[Expunge] *(Directories)*
Expunges the directory. See the section "Deletion, Expunging, and Versions in LMFS".

[Open] *(Directories)*
Opens the directory. This is the same as clicking Left on the directory name. This command is only displayed for closed directories.

[Selective open] *(Directories)*
Prompts for a wildcard name, for example, a file name containing "*" characters to indicate a wild-card component. The directory is opened and displays only those objects in the directory that match this pattern. Unspecified components default to "*". The normal [Open] command is like a [Selective open] of `*.*.*`, displaying all files. For example, if you do a [Selective open] of "`*.lisp`", only files whose type is "`lisp`" are displayed. (In this example, the version was unspecified and defaulted to "*".) The line in the display that corresponds to the directory shows this wildcard name.

[Close] *(Directories)*
Closes the directory. This is the same as clicking Middle on one of the directory's inferiors. This command is only displayed for open directories.

[Link transparencies] *(Links, directories)*
Lets you change the **:link-transparencies** of a link, or the **:default-link-transparencies** of a directory.

Each link has a property called its **:link-transparencies**. The value of this property is a list of keyword symbols. Each symbol specifies an operation to which the link is transparent. If the link is transparent to an operation, the operation is performed, on the target. If the link is not transparent to the operation, the operation is performed on the link itself. See the section "LMFS Links".

This command displays a menu showing all the operations to which a link can or cannot be transparent. Each operation to which the link actually is transparent is highlighted with reverse video. By clicking on the name of any operation, you can turn the highlighting on or off. When you are done changing the transparencies, use [Do It], and the transparencies (or default transparencies, if this is a directory) are set. You use [Abort] to abort the operation.

[Decache] *(Directories)*
When a directory is opened, the File System Editor examines the directory, sees what is there, and remembers it. If another user changes the contents of the directory while you are in the middle of editing that directory, the File System Editor does not know that anything has changed, and so what it shows you does not really correspond to the state of the file system. Using [Decache] tells the File System Editor to forget what it thinks it knows about the contents of the directory, and makes it go back to the file system to see what is really in the directory now.

[Hardcopy] *(Files)*
Hardcopies the file. Clicking on this command causes the system hardcopy menu to pop up.

[Edit File] *(Files)*
Invokes the Zmacs editor on the file.

[Load] *(Files)*
Loads the file into the Lisp world.

Two variables allow you to customize your FSEdit activity:

tv:*use-new-fsedit-command-set* *Variable*

Controls what commands are assigned to the Left and Middle mouse buttons in the File System Editor (FSEdit). **t**, the default, makes clicking Left toggle Open/Close the object and Middle Delete/Undelete the object. Setting **tv:*use-new-fsedit-command-set*** to **nil** restores the pre-Genera 8.0 behavior, where clicking Left opened an object and clicking Middle closed it.

tv:*confirm-fsedit-quick-soft-file-deletion* *Variable*

Controls whether or not deletion in FSEdit requires confirmation on a system that supports *soft* deletion, that is, where undeletion is possible. The default, **t**, is to require confirmation. Setting it to **nil** disables the confirmation. On a file system that does not have soft deletion, FSEdit always asks for confirmation.

File System Maintenance Program Commands

This section describes the menu structure of the File System Maintenance Program. File System Maintenance Program commands are organized into four levels. With each level, the potential for problems (because of incorrect use) increases.

Level 1	General file system operations.
Level 2	Local file system control operations. These are to be used by the person at your site responsible for maintaining the file system and performing backup dumps. Invoke this menu by clicking on [Local LMFS Operations] in the Level 1 menu.
Level 3	Server and maintenance operations. These should be used only by persons who are very knowledgeable about the file system. Invoke this menu by clicking on [LMFS Maintenance Operations] in the Level 2 menu.
Level 4	File system internal data structure operations. Do not attempt to use these commands unless you are an expert in internal data structures of the Lisp Machine File System. Invoke this menu by clicking on [LMFS Internal Tools] in the Level 3 menu.

Note: The commands at Levels 3 and 4 can damage your file system if used incorrectly. If you have any questions about how to use them, call Symbolics Software Support.

Some commands type out information. If the large pane is a Lisp Interaction Window, such information is simply displayed on that window. If the large pane is a File System Editor, the information is displayed in a typeout window. To flush the typeout window, press any character, or click on [Refresh Display].

Level 1 Menu

File System Operations:

[Tree Edit Root]	Enters the File System Editor, using the root directory of the local file system as the base directory. This puts the large pane into the *File System Editor* state.
[Tree Edit Any]	Enters the File System Editor; it prompts you for the name of the base directory. For information about FSEdit, see the section "Using FSEdit". This puts the large pane into the *File System Editor* state.
[Tree Edit Home Dir]	Enters the File System Editor, using your home directory as the base directory. This puts the large pane into the *File System Editor* state.

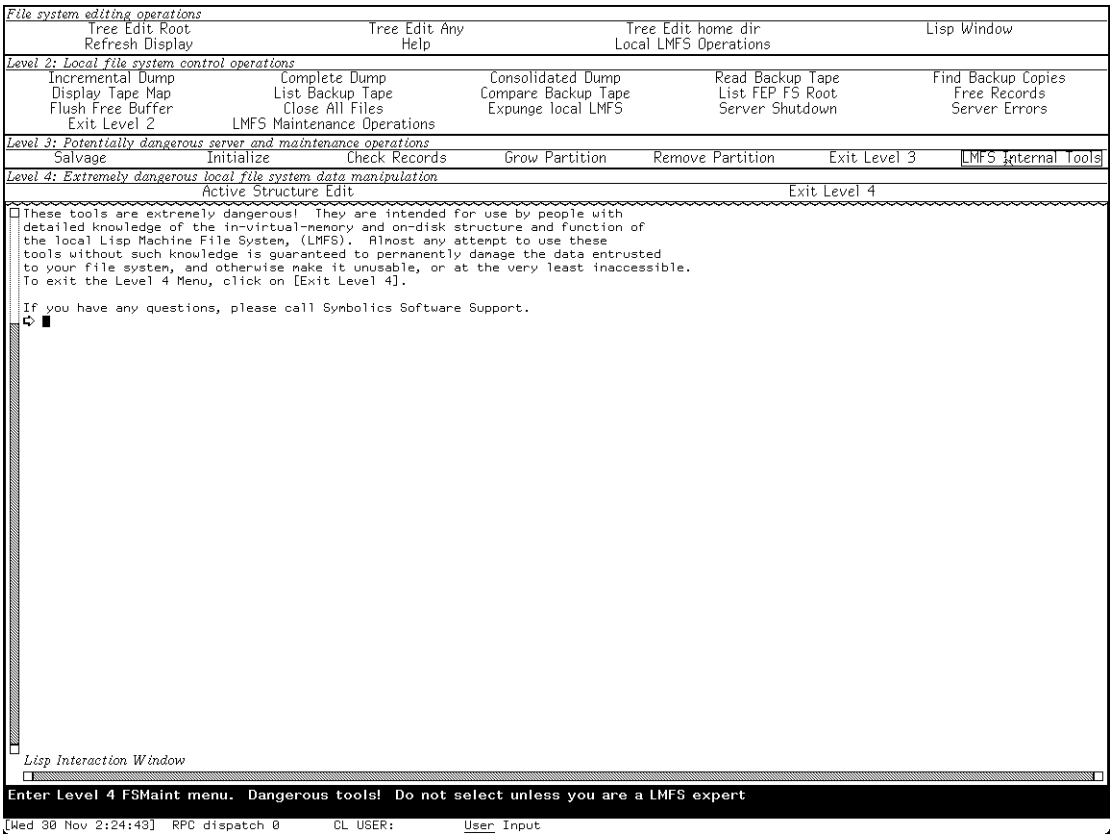


Figure 111. The Four Levels of the File System Editing Operations Menu

[Lisp Window] Puts the large pane back into the *Lisp Interaction Pane* state. This is useful for getting out of the File System Editor.

[Refresh Display] When the large pane is in the *File System Editor* state, and you use one of the commands that "types out" information, the information appears on top of the File System Editor window, and you are told "`Type any to character flush:`". You can use this command to clear the screen and redisplay the File System Editor window without removing your hand from the mouse. You can also use this command to proceed from `**MORE**` pauses.

[Help] Type out general information about the File System Maintenance program and File System Editor.

[Local LMFS Operations]
Brings up the Level 2 menu Local File System Control Operations.

Level 2 Menu

The commands in this menu are intended to be used only by the person who is responsible for maintaining the file system and performing backup dumps at your site.

Local File System Control Operations:

[Incremental Dump] Does an incremental dump of the local root directory. Offers an Accept-Values menu to adjust all parameters. See the section "Dumping, Reloading, and Retrieving LMFS Files".

[Complete Dump] Does a complete dump of the local root directory. Offers you an Accept-Values menu to adjust all parameters. See the section "Dumping, Reloading, and Retrieving LMFS Files".

[Consolidated Dump] Does a consolidated dump of the local root directory. Offers you an Accept-Values menu to adjust all parameters. See the section "Dumping, Reloading, and Retrieving LMFS Files".

[Read Backup Tape] Retrieves single files or reload full tapes. You select the activity you want from a pop-up menu. You can also use [Read Backup Tape] to list or compare tapes, if you change the default operation in the menu by clicking on [Compare] in the menu.

[Find Backup Copies] Locates files on backup tape. It prompts you for a file specification to locate.

[Display Tape Map] Displays a directory listing of what should be on a backup tape. It prompts you for the tape number. You can display a listing of any backup tape directory on any machine connected to your network by giving the machine name and the pathname of the tape directory. Standard pathname defaulting and merging work.

[List Backup Tape] Lists the contents of a backup tape that you have mounted on a tape drive. It prompts you for the tape specification.

[Compare Backup Tape]
Compares the contents of a backup tape that you have mounted to the contents of the local file system. It prompts you for the tape specification. To compare the tape to another (remote) file system, use [Read Backup Tape].

[List FEP FS Root] Lists the FEP file system's root directory from the default disk unit. See the section "Show FEP Directory Command". See the function **zl:print-disk-label**.

[Free Records] Types out information about the number of free records in the local file system. The last line tells you how many

records are marked as free, how many are marked as used, and the sum of these numbers, which is the total number of records in the file system.

Clicking Middle on [Free Records] prepares a directory-by-directory usage report of record use, indicating how many records are in use by files in each directory. It prompts you for the name of a file in which to place the report.

Clicking Right on [Free Records] displays how many records are in use in each partition. This information is necessary for the commands that allow you to change the size of, add, or remove partitions.

See the section "Free Records and the Free Record Map".

[Flush Free Buffer] Writes the internal pool of free disk records back to the disk. This happens automatically when you log out. After doing this, you can cold boot without losing records. See the section "Free Records and the Free Record Map".

[Close All Files] Calls **fs:close-all-files**. This has nothing to do with the Lisp Machine File System as such; it closes any open files in use by your machine, whether local or remote. This is occasionally useful for cleaning up after problems occur, but be aware that by using [Close All Files], you can cause new problems for any programs in the machine that are validly using files at the time.

[Expunge Local LMFS]
Expunges all directories on the local file system. It tells you how much space was recovered.

[Server Shutdown] Shuts down file servers at a future time, or reschedule or cancel a shutdown.

This command lets you shut down a file server cleanly. You run this command only on a 3600-family computer that is acting as a file server for other users. Clicking Left on [Server Shutdown] means that you plan to shut down the file system soon. It asks you for a short message to be sent to people using the file server, which you can use to explain why it is being shut down and when it will return. It also asks you when you want the shutdown to take place; the default is five minutes. All users of the file system are sent periodic messages warning them that the server is going to be shut down. Finally, when the time comes, it closes all Chaosnet servers on the machine, and disables creation of new servers. When servers are shut down, you can cold boot the machine or whatever else you want to do. While the shutdown is "in

progress" (the messages are being sent), you can cancel it by clicking Middle on [Server Shutdown] or reschedule it by clicking Right on [Server Shutdown].

[Server Shutdown] only shuts down the network server; it does not affect the local operation of the file system itself. It shuts down all servers, not just file servers, since anything that requires the file servers to be shut down also requires that all servers be shut down.

[Server Errors] Displays all the error messages associated with errors encountered by the file server. When such errors occur, you get a message that begins as follows:

```
[File Server got an error: ...]
```

The message contains descriptive information about the error.

[Exit Level 2] Returns from this menu to the top-level menu of General File System Maintenance.

[LMFS Maintenance Operations]
Brings up the Level 3 menu of Server and Maintenance Operations. **Note:** The operations on the Level 3 menu can damage your file system if misused. They should only be performed by someone who is very knowledgeable about the file system.

Level 3 Menu

The commands in this menu are intended to be used only by persons who are thoroughly familiar with file system operations.

Note: Misuse of these commands can destroy or damage the file system.

Server and Maintenance Operations (Potentially Dangerous):

[Salvage] Runs the Salvager. See the section "Running the LMFS Salvager".

[Initialize] Creates a new file system. Use this tool to add file storage space to your local Lisp Machine File System. This operation asks you several questions and prints out information to make sure you really want to initialize the FEP file that would contain the file system. These verifications are to ensure that you do not accidentally destroy any previous file system residing there. [Initialize] takes about a minute for each four thousand records (a record is four 256-word disk blocks). It queries you about the FEP file size before it initializes the new file system.

Clicking Right on [Initialize] presents a menu of initialization. This is how you can add new FEP files to the running file system. For a description of this operation: See the section "Adding a LMFS Partition".

Note: Added partitions should never be initialized. The partitions are automatically initialized by the system when they are created.

This tool is intended only for manipulating the local Lisp Machine File System, specifically, adding partitions to it for the purpose of storing files in them. If you want to create a FEP file for some other purpose, or if you want to create an additional paging file, you must do so from Lisp, using the Create FEP File command. See the section "Create FEP File Command". See the section "Increasing Available Paging (Swap) Space".

Warning: *Attempting to create an additional paging file using [Initialize] will destroy the file system on your machine, permanently and irretrievably.*

[Check Records] Checks each record in the local file system for consistency, and notify you about any problems. (This is also available from the Salvager.) This option scans the hierarchy, going through the directories, making sure that each directory entry really describes a file that agrees with it, and that each record of each file is validly identified as a part of that file.

[Grow Partition] Increases the number of blocks available in a partition. It offers you a menu to select the partition to grow and prompts for the number of blocks by which to increase the partition.

[Remove Partition] Removes active partitions from the file system and deletes them. It does this by walking over the local file system, evacuating files and directories from the partitions to be removed, to other partitions. For medium and large file systems, this operation takes a long time. In order for it to succeed, there must be enough room in other partitions to contain the evacuated files and directories. This tool determines whether or not sufficient room exists for the operation to complete successfully, and queries if it suspects that sufficient room is not available. You can click Right on [Free Records] to get a partition-by-partition report. Salvaging might be necessary to properly identify all free records. If you need to do this: See the section "Running the LMFS Salvager".

[Remove Partition] provides the option of deleting the FEP file when all LMFS files have been removed from it. If it detects that a partition is not completely empty, it reports this and allows you to abort the process.

Do not attempt to use this tool to manipulate FEP files for any other purpose, or to manipulate FEP files in use by LMFS in any other way; for example, do not use this tool on these files: lmfs.file, lmfs1.file, or fspt.fspt. Misuse of this tool causes irretrievable destruction of data in your file system.

[Exit Level 3] Return from this menu to the Level 2 menu of Local File System Control Operations.

[LMFS Internal Tools]
Bring up the Level 4 menu of File System Data Manipulation Operations.

Warning: *Editing the internal structures of your file system can result in data being irretrievably lost. You should not use these tools unless you are sure you know what you are doing.*

Level 4 Menu

The commands in this menu are intended only for use by persons who are thoroughly familiar with the internal organization and implementation of the file system.

Note: Misuse of these operations can destroy or damage the Lisp Machine File System.

Local File System Data Manipulation (Extremely Dangerous):

[Active Structure Edit]
Edits the active file system data structure. This displays "active" internal data structure as a scroll window, and is intended to be used by those debugging local file system problems.

[Exit Level 4] Returns from this menu to the Level 3 menu of Server and Maintenance Operations.

Access Control Lists

Introduction to Access Control

The Site Administrator can configure a Symbolics Lisp Machine file server to provide file protection for its LMFS and FEP file system files. The file protection mechanisms then control access to files on the protected server machine. They *do* protect files from network file users, but they *do not* protect files from users with physical access to the file server's console. They *will* protect against mistakes, but they *will not* protect against malicious users.

There are two steps to protecting files in a LMFS file system using the file protection mechanism:

1. Decide what controls you, as owner, want to impose on access to directories. There are six methods of access control available. Details for each level are given in another section. See the section "Access Control Model: What You Can and Cannot Protect".

2. Configure the server machine. There are three steps involved in configuring a file server. These steps need only be performed once. See the section "Configuring a File Server".

Access Control Model: What You Can and Cannot Protect

The file protection mechanism offers a number of ways to protect your files from others.

1. **Genera can protect against network users**. Genera can protect local files against inter-machine access. At this level of control, users accessing files across the network through the standard protocols cannot run arbitrary Lisp programs or access arbitrary data. The code that implements the network servers represents a *reference monitor*, in the sense that all user access to files passes through it. Therefore, a properly configured Symbolics file server can provide a reasonable level of assurance that the file protection mechanisms do indeed protect files. The site administrator can check this with the file server log described later. See the section "The File Server Activity".

   Genera has no software architecture to support intra-machine file access protection. The flexible programming environment allows any user program to access all data on the machine. No computer data security can assure that it is resistant to penetration unless the architecture isolates the user and the user's programs from the data structures that define and implement system software and, in particular, its security mechanisms. Therefore, the ACL mechanism makes no attempt to protect files from a local user.

2. **The file server can require login**. In configuring a file server, the Site Administrator can choose whether to require login (identification and authentication) of all users of the file server. Login is implemented via user names and passwords. If the Site Administrator does not require login, anyone can establish a file server connection to the server and access any files that are marked as accessible to everyone. If the file server does require login, all users must supply their name and password before they are allowed to access the file server. See the section "Administering Names, Capabilities, and Passwords".

   Supplying a name and password:

- Demonstrates that the user is an authorized user of the facility, and thereby protects all the data on the server from unauthorized access.

- Allows the site administrator to create and maintain an audit trail of the user's access via log files.

Note: An untrusted host is always required to log in and *can never* access a file whose ACL is empty.

See the section "The Secure Subnets Site Attribute".

3. **The site administrator can define capabilities**. The file protection model allows the owner of a directory to grant or deny access to data in that directory to individuals on the basis of the user name they supply at login time.

   In addition, the Site Administrator can define capabilities. A capability is an access identity shared by multiple users that describes some common access privileges. For example, the site administrator might wish to specify that all of the people on a particular project (the Vision project, for example) have access to a set of files. Rather than listing all the people on the project (a list that might change over time) when specifying the access, the administrator can grant access to the Vision capability and then grant all of the appropriate users the right to use the capability. The site administrator grants a user the right to use a capability by assigning a password for the capability. Each user of a capability has a different password for the capability. Each user has multiple passwords: one for the user name, and one for each capability the user has the right to use.

   Users can utilize a capability by turning it on with the Enable Capabilities command. It prompts for a password, then turns on that particular capability for the requesting user.

4. **The owner can specify which users and capabilities can access the files**. The owner of a LMFS file can control access to files and directories by using the Edit ACL command to edit the Access Control List (ACL) associated with each directory. See the section "Edit ACL Command". The ACL for a directory controls access to the directory and all files within the directory. An access control list consists of an ordered set of pairs. Each pair consists of an access name (a user name or a capability name) and a list of access modes. The modes are:

Mode	*The ability to*
:read	read, probe, or perform **fs:file-properties** on files
:write	modify existing files

:append	append to existing files
:properties	delete files, expunge files, or change properties of files
:list	list the file names in the directory
:supersede	create new versions of existing files
:create	create new files (not new versions)
:owner	change the directory's ACL and other properties

The user's access to a file or directory is determined by looking through the list of access names for that file or directory. The system determines a user's right to perform an operation by checking the user name and any enabled capabilities against the ACL. Whichever currently enabled access name first matches a name in the ACL determines the user's access. The reserved user name "*" in an ACL matches all access names.

New directories are initialized with an ACL copied from their parent directory. For an example of an ACL: See the section "Edit ACL Command". **The file server initialization determines the interpretation of an empty ACL**. In configuring the file server, the Site Administrator chooses whether an empty ACL grants access to all users or to no users. Granting access to all from an empty ACL is called *permissive access*.

Note: An untrusted host can *never* access a file with an empy ACL.

5. **The file server can control access to the FEP file system**. There are no ACLs on individual FEP file system directories. There is a single directory in the LMFS hierarchy that controls access to all files and directories in the FEP file system. Note that when ACLs are turned on, the command Show FEP Directory does not work unless the **:write**, **:read** and **:list** modes are enabled for the directory.

   LOCAL:>File-Server>FEP-File-System-Access>

6. **The site administrator controls access to the physical console**. This protection scheme does not protect files from users with physical access to the file server's console. Therefore, the Site Administrator must consider some method of controlling physical access to all secure consoles.

Configuring a File Server

Configuration of a file server requires three steps that are performed only once.

1. Create the necessary directories. The function **fs:setup-file-server** creates several directories required for the operation of a file server, with or without security. These are:

Directory	*Contents*
>File-Server>	Password files
>File-Server>Server-Logs>	File server log files
>File-Server>FEP-File-System-Access>	The ACL for this directory is the protection for the FEP file system

2. Set the access on system directories. Use the Edit ACL command to set appropriate access for all of the directories listed above. It is *very* important that the administrator set the protection for these directories; otherwise, this protection scheme can be easily broken. Note that the command Show FEP Directory does not work unless the **:write**, **:read** and **:list** modes are enabled for the directory >File-Server>FEP-File-System-Access>.

3. Set up the file server init file. The file server init file should contain an invocation of the function **fs:initialize-secure-server**. There is an example of this in the file SYS:EXAMPLES;FILE-SERVER-INIT-FILE.TEXT.

Administering Names, Capabilities, and Passwords

The database of names, capabilities, and passwords is kept in the file

>File-Server>passwords.data.

This file is encrypted. To define a user name, or to change a user's password, or to change a user's password for a capability, use the Set Password command in the File Server activity.

To make a user name undefined, or to revoke a user's right to a capability, use the Remove Password command in the File Server activity.

The File Server Activity

The File Server activity includes commands for setting and removing passwords, logging file system activity and errors, and displaying file server status. The activity uses a frame accessible via the CP command Select Activity, using the File Server argument. The frame is divided into five smaller panes. File Server commands can be entered through typein at the bottom pane, or by clicking on commands available in the File Server Command pane.

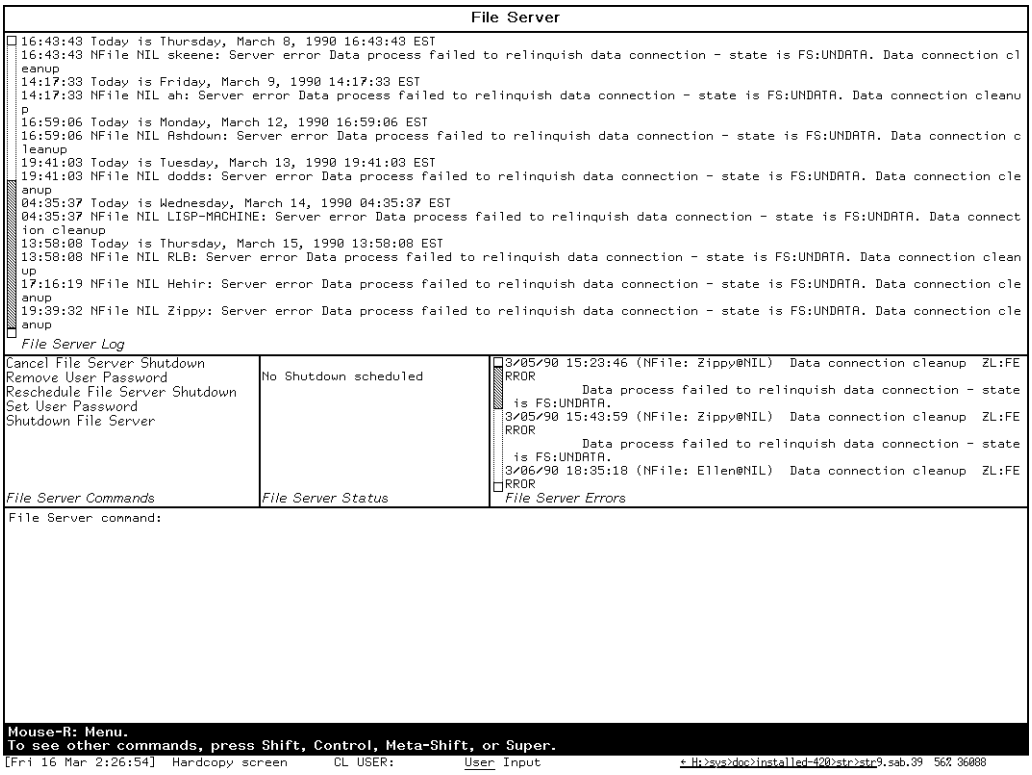


Figure 112. File Server Activity Window

There are a few commands available in the File Server Command pane. These are:

[Cancel File Server Shutdown]
: Cancels an already scheduled File Server shutdown.

[Remove User Password]
: Removes the password for a user, or remove the password for a capability for a user. The command prompts for the user's name.

[Reschedule File Server Shutdown]
: Changes the time for a scheduled shutdown. The command prompts for a new number of minutes to shutdown and a string that is used for logging purposes.

[Set User Password]
: Sets the password for a user, or for a capability for a user. The command prompts for the user's name.

[Shutdown File Server]

Schedules a shutdown of the File Server Activity. The command prompts for the number of minutes to shutdown, and for a string. The string is used in the log file, and can be used for logging the reason for the shutdown.

The File Server Status pane shows the current status of the File Server Activity. Currently this pane only displays whether a shutdown is scheduled, and if so, how many minutes until the shutdown occurs.

The File Server Errors pane shows any errors that have occurred. This includes all errors specified to the **:log-error-response-flavors** keyword argument to **fs:initialize-secure-server**, such as an unrecognized user trying to log in.

The File Server Log pane shows any File Server accesses that have occurred. This includes all messages specified to the **:logging-keywords** keyword argument to **fs:initialize-secure-server**, such as users logging in and out.

ACL-Related Lisp Functions

The functions **fs:initialize-secure-server** and **fs:setup-file-server** are initialization forms used by the file servers to create and maintain the access lists.

fs:initialize-secure-server &key *access-permissive* (*login-required* **t**) (*permitted-services* **fs:*trusted-services***) (*logging-keywords* **fs:*default-secure-server-logging-keywords***) (*log-error-response-flavors* **fs:*default-secure-server-error-response-log-flavors***) (*lmfs-fspt-pathname* **lmfs:*fspt-pathname***) *Function*

Starts up the security functions for the file server. The function takes the following keyword arguments. This command should be executed from the file server init file.

:access-permissive When the value is **t**, an empty ACL gives access to all users. When the value is **nil**, an empty ACL gives access to no users.

:login-required Flags whether or not login is required to access files. When the value is **t** (the default), a login is required.

:permitted-services A list of network service keyword names, such as **:tcp-ftp**, **:name**, and **:chaos-status**. When running a secure server, services that permit a user to get arbitrary access to the server machine should not be enabled. In particular, the **:eval** and **:login** services should not be enabled. The variable **fs:*trusted-services*** contains the services that Symbolics recommends be enabled.

:logging-keywords These keywords control what information is put into the file server log. The currently defined keywords are:

:login Logs a message for each login and logout.

:server-error-response
Logs a message whenever an error condition in the server is translated into an error response to the user, subject to **:log-error-response-flavors**.

:untrusted-transaction
Logs all transactions involving users who are not on a trusted subnet.

:server-bug-report Logs a message each time an automatic bug report is sent.

:server-error Logs a message every time the server takes an error that is not just a user error.

:log-error-response-flavors
A list of error conditions that should be logged. Only those errors that satisfy **typep** of a flavor in this list are logged.

:lmfs-fspt-pathname
The pathname of the LMFS File System Partition Table for this file server. It is of the form fep*n*:>fspt.fspt, where *n* is the disk unit where the fspt file resides.

fs:setup-file-server *Function*

Creates the directories that are required for the operation of a file server, with or without security. **fs:setup-file-server** should be executed when setting up a file server. It creates the following directories required for the operation of a file server:

Directory	*Contents*
>File-Server>	Password files
>File-Server>Server-Logs>	Log files
>File-Server>FEP-File-System-Access>	The ACL for this directory is the protection for the FEP file system

ACL-Related Command Processor (CP) Commands

The Command Processor (CP) Enable Capabilities and Disable Capabilities commands enable users to access groups of files. The Command Processor (CP) Set Password and Remove Password commands enable administrators to control access to capabilities. Finally, the Command Processor (CP) Edit ACL command enables users to set the access for directories.

Enable Capabilities Command

Enable Capabilities *host capabilities*

Turns on specified *capabilities* for *host* after checking access requirements. The host prompts for your password for *capabilities*.

host	The name of the host on which you want to enable *capabilities*.
capabilities	One or more of the capabilities available for *host*. Each specified capability must be already recognized by the server before access can be enabled.

Disable Capabilities Command

Disable Capabilities *host capabilities*

Turns off specified *capabilities* for *host*.

host	The name of the host on which you want to disable *capabilities*.
capabilities	The capabilities you want to disable on *host*.

Set Password Command

Set Password *user-name capability*

Sets the password for *capability* for *user-name*. This command must be executed from the File Server activity on the console of the host that knows about *capability*. This should be done by the Site Administrator responsible for assigning passwords.

user-name	The name of the user for whom you want to set a password.
capability	A capability that *user-name* can have access to.

`Set Password` *user-name* `None`

sets the general access password for *user-name*.

Remove Password Command

Remove Password *user-name* &optional *capability*

Removes either the password for *user-name*, or the password for *capability* for *user-name*. This command must be executed from the File Server activity on the console of the host that knows about *capability*. This should be done by the site administrator responsible for assigning passwords.

user-name The name of the user for whom you want to remove a password.

capability A capability that *user-name* can no longer have access to.

Edit ACL Command

Edit ACL *directory*

Starts up a small window, editing the Access Control List for *directory*. It is used to create, edit or remove access control lists for LMFS directories. This command can be executed from any console, by the owner of *directory*. This should be coordinated with the Site Administrator, so that passwords can be assigned and any new capabilities can be set up. See the section "Access Control Model: What You Can and Cannot Protect".

The window is divided into four panes: the top displays the name of the directory, the next pane displays the existing ACLs, next is a command pane, and the bottom pane is a minibuffer for typein.

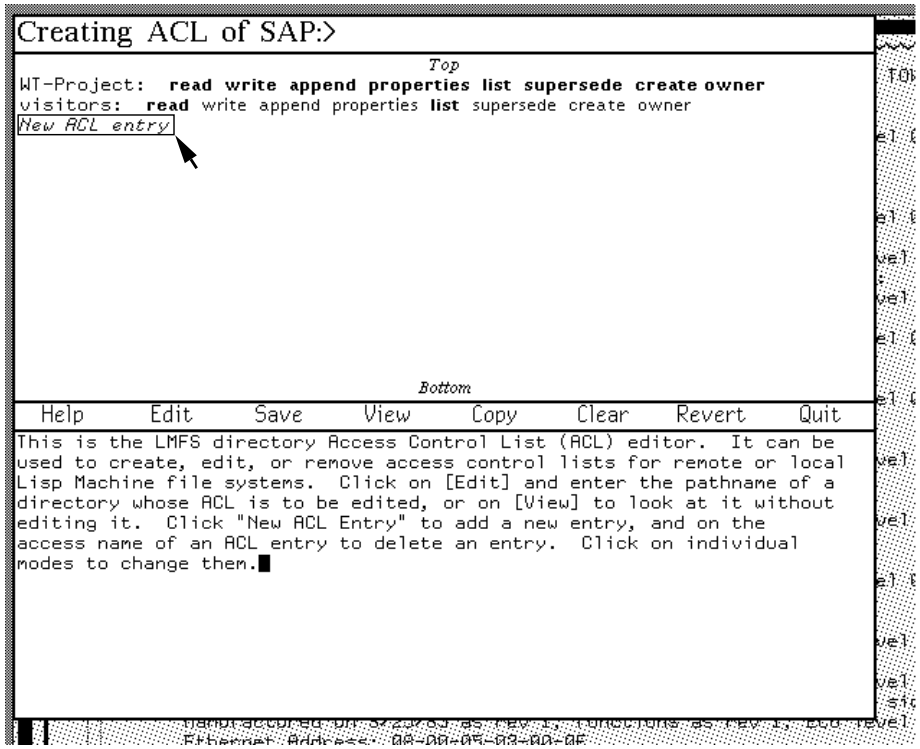


Figure 113. Access Control List Editor

[Help] Display general information about the Access Control List editor.

[Edit] Prompts for the name of a directory, and starts an edit of its ACL. If there is no ACL for that directory, the second pane will only contain *New ACL entry*. Click on this command to add a new ACL. To delete an entry, click on the access name of that entry.

Each entry consists of an access name, and a list of modes that can be enabled or disabled. Clicking on a mode toggles whether or not it is enabled. Modes that are enabled appear in boldface type, and modes that are disabled appear in roman type.

[Save] Installs the permissions for the host you are editing by changing the file properties for that directory.

[View] Prompts for the name of a directory, and displays its ACL. This is done in read-only mode, so you cannot modify the ACL using this command.

[Copy] Creates a new ACL for a directory by copying the current ACL.

[Clear] Disables all modes for the current ACL entry.

[Revert] Reverts the ACL to the modes specified in the saved file for this directory. If the ACL has never been saved, the editor asks if you want to discard the current ACL information. When you confirm, all entries are discarded.

[Quit] Return from the Edit ACL command to whatever you were previously doing.

For the example here, the capability named "WT-Project" allows all operations on the directory SAP:>. The capability named "visitors" only allows the reading and listing of files in the directory.

Managing and Troubleshooting the LMFS

The Create Directory, Delete Directory, Expunge Directory, Show Directory, and Delete File Command Processor (CP) commands and the Lisp function **lmfs:fix-file** described within this section are useful for Site Administrators who manage and troubleshoot LMFS file systems.

For related information,

- See the section "Salvager-Related Lisp Functions".
- See the section "ACL-Related Lisp Functions".
- See the section "ACL-Related Command Processor (CP) Commands".

Create Directory **Command**

Create Directory *pathname*

Creates a directory for storage of files on a file system specified in *pathname*.

pathname The pathname of the directory to be created.

Delete Directory **Command**

Delete Directory *pathname keywords*

Deletes and expunges the directory represented by *pathname*.

pathname The physical name of the directory to delete. (Using a logical pathname could result in an ambiguous directory reference.) Delete Directory asks you to confirm the deletion before doing it.

You should make sure that *pathname* represents a directory. `A:>KJones>projects>` is a directory, and

```
Delete Directory A:>KJones>projects>
```

offers to delete the `projects` subdirectory. However,

```
Delete Directory A:>KJones>projects
```

is a file name whose directory component is >KJones>; it therefore offers to delete the directory `KJones`. Similarly, the pathname

```
A:>KJones>projects.directory
```

is a file representation of a directory. It is not the directory name for use with commands that manipulate files in directories, so giving it as an argument to Delete Directory also offers to delete the directory `KJones`.

keywords :Confirm

:Confirm {Yes, No, Each} Whether to ask before deleting each subdirectory. The default is Yes. The mentioned default is Each.

Expunge Directory **Command**

Expunge Directory *pathname keywords*

Expunges files marked for deletion.

pathname The pathname of the directory to be expunged. The default is the usual file default.

keywords :Query

:Query {Yes, No, Ask} Ask for confirmation before expunging the directory. The default is No. The mentioned default is Yes. :Query is useful when *pathname* contains wildcard.

Show Directory **Command**

Show Directory *pathname keywords*

Displays a directory listing. The default for name, type, and version of *pathname* is **:wild**. The format of the listing varies with the operating system. For more information, see the section "Examining the Lisp Machine File System" and see the section "Interpreting Directory Listings in FSEdit".

pathname The pathname of the directory to list. The default is the usual file default.

keywords :Author, :Before, :Excluding, :More Processing, :Order, :Output Destination, :Since, :Size

:Author {*user-id*} Show those files written by this user.

:Before {*date*} Show only those files created prior to this date.

:Excluding {*file1, file2* ...} Exclude files that do not match the indicated wildcard filenames from the directory listing. Logical pathnames and pathnames having versions specifiers of "oldest" and "newest" are not permitted as excluded *files* because the exclusion test will be done against the truename of the physical pathname, and such pathnames would never match.

:More Processing {Default, Yes, No} Controls whether **More** processing at end of page is enabled during the output to interactive streams. The default is Default.

If No, output from this command is not subject to **More** processing.

If Default, output from this command is subject to the prevailing setting of **More** processing for the window. (See the section "FUNCTION M".)

If Yes, output from this command is subject to **More** processing unless it was disabled globally (see the section "FUNCTION M").

:Order {oldest-first, smallest-first, largest-first, newest-first, standard} Show files in this order. The default is standard, which is usually alphabetical.

:Output Destination
{Buffer, File, Kill Ring, Printer, Stream, Window} Where to redirect the typeout done by this command. The default is the stream ***standard-output***.

:Since {*date*} Show only those files created after this date.

:Size {*integer*} Show only those files the same size or larger than *integer* blocks. The default is 0, meaning that all files will be listed, even if they are empty.

Delete File **Command**

Delete File *pathname keywords*

Deletes or marks for deletion the file *pathname*.

pathname Pathname of the file to delete. The default is the usual file default. The version defaults to newest.

keywords :Expunge, :More Processing, :Output Destination, :Query

:Expunge {Yes, No, Ask}. Whether to expunge the file. The default is No. The mentioned default is Yes.

:More Processing {Default, Yes, No} Controls whether **More** processing at end of page is enabled during the output to interactive streams. The default is Default.

If No, output from this command is not subject to **More** processing.

If Default, output from this command is subject to the prevailing setting of **More** processing for the window. (See the section "FUNCTION M".)

If Yes, output from this command is subject to **More** processing unless it was disabled globally (see the section "FUNCTION M").

:Output Destination
{Buffer, File, Kill Ring, Printer, Stream, Window} Where to redirect the typeout done by this command. The default is the stream ***standard-output***.

:Query {Yes, No, Ask} Whether to ask for confirmation before deleting the file. The default is No. The mentioned default is Yes.

lmfs:fix-file *pathname* *Function*

An ECC error in a LMFS partition is reported by the Debugger and looks like this:

```
>>Error: %DISK-ERROR-ECC during a %DCW-READ32 on unit 3.,
    Fatal ECC error,
    4. pending transfers associated with this disk event aborted.
    Word 0 of file header, record #o161470 of partition 3
         (File partition in FEP3:>lmfs3.file.1)
         For T:>sam>hacks>definitions.lisp.12
```

The recommended treatment for this error is to run **lmfs:fix-file** on the machine that is getting the error. If the problem is in a directory, you should run the salvager, since some files may have been orphaned.

A directory is a file that contains an entry for each file residing in that directory. If the directory is damaged, entries may be lost. The corresponding files still exist, but do not appear in the directory listing. These files are considered orphans, and are often recoverable by the salvager.

lmfs:fix-file is a potentially dangerous tool and should be used with caution. It notifies the user of problems encountered and asks before taking corrective action, but it can destroy the file it is applied to. **Note:** If you do not understand the questions being asked or would like assistance with the use of this tool, please contact Symbolics Software Services.

Run **lmfs:fix-file** using the fully specified pathname mentioned in the error message:

```
(lmfs:fix-file "T:>sam>hacks>definitions.lisp.12")
```

lmfs:fix-file mentions problems it finds and queries the user before taking corrective action.

Symbolics Store-and-Forward Mailer

Overview of the Mailer

The Mailer is a program that provides mail forwarding and delivery services to users at a site. The Mailer is distinct from Zmail, which is a user program for reading and composing mail.

The Mailer program runs on a Symbolics machine designated as a *mail server*. The mail server host must also be a file server. A large site that has more than one file server might also have more than one mail server.

Like other network services, the mail service is invisible to users. The Mailer does its work automatically. For information about installing the Mailer, see the section "Installing and Configuring the Mailer".

This is a *store-and-forward* mailer, meaning that when a network connection cannot be made immediately to the receiving machine, the mail server saves the mail and retries the transmission until it is successful. In other words, store-and-forward mailers guarantee, within reason, that mail transmissions are reliable regardless of the state of the network when a user sends or replies to a message.

The Mailer provides the following features:

- Delivery of mail to user inboxes
- Local delivery of mail to archive files
- Special handling of hardcopy mail (for users who prefer their mail in printed form)
- Support for mailing lists
- Transaction logging

Administrative Tasks Associated with the Mailer

After the initial installation, the Mailer usually runs without any need for intervention. The most common administrative tasks include informing the mailer of a new user who will receive mail, adding a new user to various mailing lists, and creating new mailing lists; these all fit in the category of updating the mailbox table. From time to time, you might find it useful to specify options to customize the Mailer. At a site where some users receive their mail in hardcopy form only, the system administrator periodically gives a command to hardcopy the mail.

These tasks involve editing the configuration files and using commands in the Mailer Operations window.

See the section "Mailer Operations Window".
See the section "Mailer Operations Commands".

As system administrator, you might find it useful to get information on what the Mailer is doing. You will also have to trouble-shoot the Mailer if anything goes wrong.

Installation
: See the section "Installing and Configuring the Mailer".

Updating the mailbox table
: To add a new user, or make any changes to mailing lists, you need to edit the file named `>Mail>Static>Mailboxes.text` and save it. See the section "A Sample Mailboxes.Text File". To make the changes take effect immediately, use the Update Mailbox Table command in the Mailer Operations window. (At periodic intervals, the Mailer updates the mailbox table itself.)

Customizing the Mailer
: Several aspects of the Mailer can be configured by setting variables in the `Options.lisp` file. For information on the options, see the section "Mailer Options". To set one of the options, edit the file `>Mail>Static>Options.lisp` and save the file. To make the changes take effect, use the Update Options command in the Mailer Operations window.

Hardcopying Mail for Users

Some users receive mail in hardcopy form only; these users are designated in a `deliver-hardcopy` form in the `>Mail>Static>Mailboxes.text` file. See the section "A Sample Mailboxes.Text File". To hardcopy the mail for these users, use the Hardcopy Mail command in the Mailer Operations window.

Getting Information on the Mailer

The Mailer Operations window displays information on what the Mailer is doing; specifically, it shows the log file as it changes. It also shows the status of Mailer processes. Several commands in the Mailer Operations window enable you to ask for more specific information, or to cause the Mailer to perform one of its normal operations, such as trying to send queued mail to a host which has not been responding. See the section "Mailer Operations Commands".

Trouble-shooting the Mailer

See the section "Trouble-shooting a Mailer Crash".

Files and Directories Used by the Mailer

The Mailer operates by using several files stored on the file system (LMFS) of its server machine. There are three directories used by the mailer.

The `>Mail>Static>` directory contains files that customize the Mailer for your site, and are prepared by the system administrator during the Mailer installation procedure. The system administrator will occasionally edit one of these files to create new mailing lists or update existing ones. The Mailer also writes log files in this directory.

The `>Mail>Dynamic>` directory contains files that are read and written by the Mailer only. The Mailer writes mail to files in this directory.

The `>Mail>Hardcopy>` directory contains files that are read and written by the Mailer only. The Mailer writes hardcopy mail to files in this directory.

Files in the >Mail>Static> Directory

`>Mail>Static>Options.lisp`

This file contains options that customize the Mailer's operation for your site. This file is set up by the system administrator when installing the Mailer. For an example, see the section "A Sample Options.Lisp File". For information on the options, see the section "Mailer Options".

The `Options.lisp` file is stored on each mailer host so that the Mailer can be initialized even if the SYS host is unavailable. The Mailer reads this file when it is started. If you edit this file, use the Update Options command in the Mailer Opera-

tions window to make the Mailer read the new version of the file

`>Mail>Static>Mailboxes.text`

This file defines mailing lists and delivery paths for mail handled by this server. This file is set up by the system administrator when installing the Mailer. If you edit this file, use the Update Mailbox Table command in the Mailer Operations window to make the Mailer read the new version of the file. For more information, see the section "A Sample Mailboxes.Text File".

`>`*date*`.log` These are log files, which contain information on mailer events.

Files in the >Mail>Dynamic> Directory

`>Mail>Dynamic>Forwarding.text`

This is a forwarding file which is written by the Mailer. This file is used at sites with multiple mail servers. One mail server sets up a list of hosts for which it should write forwarding tables. It writes these forwarding tables whenever its Mailer is booted, or whenever its `Mailboxes.text` file changes. See the section "Configuring Large Sites for Multiple Mail Servers".

`>Mail>Dynamic>*.Mail`

These files hold actual messages being processed.

`>Mail>Dynamic>*.work`

These files are working files created and managed by the Mailer. One is created for each `*.mail` file. These files record information about the message's recipients, and they are processed in such a way as to prevent a message from being sent twice to the same recipient if the Mailer fails during a transmission to several hosts.

`>Mail>Dynamic>Mailboxes.Snapshot`

After each successful reading and "compilation" of the mailboxes table, the Mailer writes an image of the table into this snapshot file. At cold boot time, the Mailer tries to read `Mailboxes.Snapshot` first. If, for any reason, this fails, the Mailer reads and "compiles" `Mailboxes.text` and, if present, `Forwarding.text`.

Since the binary image is always written after sucessful "compilations" of the mailboxes table, it always reflects the latest working state of the mailboxes table and, therefore, is always the proper file to use when cold booting. If `Mailboxes.text` and/or `Forwarding.text` is changed during the interval between the last "compilation" and a cold boot, the Mailer notices this

fact immediately once in operation and performs its normal background update of the mailboxes table.

Files in the >Mail>Hardcopy Directory

`>Mail>Hardcopy>`*user*`.Mail`

These files contain mail that is meant to be hardcopied only, by using the Hardcopy Mail command in the Mailer Operations window.

Mailer Options

You set Mailer options in the `>Mail>Static>Options.lisp` file. Most of the Mailer options are variables, which you can set with **setf** forms. You can examine the values of these variables by evaluating them. Two Mailer options are not variables; they are functions that you can call in the `Options.lisp` file.

Note that whenever the Mailer is started (when services are enabled, or when the Start Mailer command is used) all mailer options are reset according to the forms in the `>Mail>Static>Options.lisp` file. Any variables that are not set in the `Options` file are reset to to their default values.

If you edit this file, use the Update Options command in the Mailer Operations window to make the Mailer read the new version of the file.

mailer:network-bad-gateways *Variable*

A list of hosts and/or (host network) pairs that should be avoided as gateways. All host and network names are strings. The default is **nil**.

mailer:*log-file-generation-retention-count* *Variable*

An integer that specifies how many log files should be kept. The default is 10.

mailer:*log-file-maximum-size* *Variable*

An integer that controls the maximum size of Mailer log files. The default is 450K characters. When this size is reached, the Mailer automatically creates a new log and writes data to it.

mailer:hardcopy-mail-retention-count *Variable*

A number giving the generations of old hardcopy mail to keep. The default is 3. A "generation" of hardcopy mail is defined each time you use the Hardcopy Mail command. Generation one is mail that has not yet been hardcopied; generation two is mail that was hardcopied the last time Hardcopy Mail was used, and so on.

mailer:deferred-delivery-times *Variable*

A value that specifies how to handle "deferred" mail (for instance, mail that you allow to accumulate for transmission with Dialnet, during a single phone session). **nil** means "never initiate deliveries; wait for the connection to be opened from the other end." **t** means "initiate delivery immediately; deliver mail as soon as possible". You can specify an interval, such as "8 hours", meaning to deliver deferred mail at that interval. Finally, you can specify a list of times, for deferred delivery, for example:

```
'("10:10pm" "6:45am" "2:30pm")
```

The default is **nil**.

mailer:deferred-receipt-hosts *Variable*

A list of host objects to probe for incoming mail. The default is **nil**.

mailer:deferred-receipt-times *Variable*

A value that specifies when to probe deferred-receipt hosts. **nil** means never, a time interval specifies how often, or a list of times gives specific times of day, as with **mailer:deferred-delivery-times**. The default is **nil**.

net:notify-hosts *Option*

A list of hosts (symbols or strings or host objects) to notify when Mailer errors occur. The default is **nil**.

mailer:failed-mail-reply-mail *Variable*

A value other than **nil** means to send a message about failed mail to the mail's originator. Here is an example of such a message:

```
Date: Friday, 16 June 1990, 15:31-EDT
From: Postmaster at ACME.CSNET.COM
Subject: Unable to deliver letter
Message-ID: <900316153146.5.FILE-SERVER@ACME.CSNET.COM>
To: KJones@WOMBAT

Unable to deliver letter to the following recipient:
  "A:>JSmith>mail.text"@ACME: No such address.

----- Text of letter follows -----
...
```

The default is **t**.

mailer:failed-mail-reply-file *Variable*

A value other than **nil** means to copy failed mail to a special log file. The default is **nil**.

mailer:forwarding-table-hosts *Variable*

A list of hosts to which forwarding tables should be written. This is used only in sites that use muliple mail servers. See the section "Configuring Large Sites for Multiple Mail Servers".

Note that changes to this variable don't take effect until the following form is evaluated:

```
(mailer:clear-parsed-forwarding-table-hosts)
```

If you put the above form into the Options.lisp file after the form which sets **mailer:forwarding-table-hosts**, then whenever the Mailer reads Options.lisp (e.g., because you clicked on "Update Options" in the Mailer log window), the new list of hosts will take effect.

mailer:enable-slow-delivery-process *Variable*

The Mailer maintains a list of what it considers "slow networks". Presently, the only slow network is Dialnet.

If the value of this variable is **t**, a special "slow delivery" process is created when the Mailer starts up; this process delivers messages to hosts that are directly reachable but only via slow networks. If the value of **mailer:enable-slow-delivery-process** is **nil**, the foreground process performs delivery to directly reachable hosts over slow networks, as well as local delivery and (possibly, depending on the value of enable-delivery-processes) indirectly reachable hosts, or hosts reachable over "fast" networks.

The default is **t**.

mailer:enable-delivery-processes *Variable*

If the value of **mailer:enable-delivery-processes** is **t**:

- All foreign deliveries (except possibly direct, slow ones) are handed off to separate "delivery processes", and the foreground process performs local deliveries only.

- When the Mailer starts up, a number of delivery processes (the number being taken from the variable **mailer:number-of-delivery-processes**) are created.

If the value of **mailer:enable-delivery-processes** is **nil**:

- The foreground process performs both local deliveries and all foreign deliveries (except possibly direct, slow ones) to fast networks.

- The variable **mailer:number-of-delivery-processes** is ignored.

The default is **t**.

mailer:number-of-delivery-processes *Variable*

The number of delivery processes which should be started up to handle delivery of foreign mail. Deliveries of local mail are still performed by the foreground process, and are not affected by this variable. The default is 1. You can set it to a higher number for a mail server that processes a great volume of mail, to devote more of the machine's resources to delivery of foreign mail.

This variable works in conjunction with **mailer:enable-delivery-processes**.

mailer:define-host-recipient-processing-time *host processing-time* *Function*

The Mailer has a timeout when sending a message to a host. By default, this timeout is based on the length of a message. Sometimes, for a heavily loaded or slow host, short messages with many recipients will not be finished before the timeout. If you notice this happening in the log file, you can call this function in the `Options.lisp` file to control the timeout.

This mailer option causes the timeout for the given *host* to be based on the number of recipients of a message. The timeout waits *processing-time* for each recipient of the message.

host is the name of a mail server host. *processing-time* is a period of time. For example:

```
(define-host-recipient-processing-time "Pegasus" "15 seconds")
```

The default is one second.

mailer:define-host-user-id *host host-user-id password* *Function*

If your mail server needs to write forwarding tables to a host that does not allow anonymous logins, you should call this function on the Mailer server to specify for that host a user id and password to be used.

host is the name of the remote host which disallows anonymous logins. *host-user-id* is a string specifying a user id, and *password* is a string specifying the password for that user id.

mailer:undeliverable-host-interval *Variable*

The minimum interval before the mailer gives up on delivering a message when mail cannot be delivered to a given host. The default is one week.

mailer:unresolvable-host-interval *Variable*

The minimum interval before the mailer gives up on delivering a message when mail cannot be delivered to a given host because domain information on it cannot be resolved. The default is one week.

mailer:logs-directory *Variable*

The pathname of the directory where the Mailer will store its log files. The default is **#P"LOCAL:>Mail>Dynamic>"**.

mailer:probe-interval *Variable*

The minimum time to wait before retrying a host that was down when delivering outgoing mail. If this value is 10 minutes, the first probe will occur after 10 minutes, the second 20 minutes after the first, the third 30 minutes after the second, etc. The default is 10 minutes.

mailer:mailbox-deletion-threshold *Variable*

If the number of entries in the mailbox table decreases by more than the threshold amount specified by this option, the mailer will refuse to perform any background updates on the theory that a major editing mistake has taken place. This option is interpreted as follows:

nil	Don't check for shrinkage.
0 < *threshold* < **1**	Refuse if more than this percentage of entries would be deleted.
threshold ≥ **10**	Refuse if more than this number of entries would be deleted.

The default is 100.

A Sample Mailboxes.Text File

Here we show a sample `>Mail>Statice>Mailboxes.text` file, and then describe the contents of the file. This file is appropriate for sites that use only one mail server; for information on sites that use multiple mail servers, see the section "Configuring Large Sites for Multiple Mail Servers".

```
;; -*- Mode: Lisp; Package: Mailer; Syntax: Common-Lisp -*-

;; This file belongs on White only.

(define-local Postmaster Susan)

;;; People who get mail delivered on White
(deliver-local Robert Susan Peterson Smiley)
```

```
;;;These people get mail in hardcopy only
(deliver-hardcopy Marilyn Steve)

;;; Give Susan the alias Sullivan
(define Sullivan Susan)

;;; The basic mailing list for bugs.  Each person listed
;;; gets a copy of each bug report, and a copy goes to the files.
(define Bug-Genera
    (list Susan Robert Smiley
            ">Mail>Archive>Bug-Genera.text"))

;;; Useful to have a mailing list that allows you to reach everyone
(define everybody
    (list Robert Susan Peterson
          Smiley Marilyn Steve Nelson))

(define sports-followers
        (list Smiley Marilyn Robert)
```

These forms are not evaluated in the normal Lisp way. The Mailer evaluates them specially. You should use the syntax as used in this sample file, and should not, for example, replace `(list x y z)` with `'(x y z)`.

Here are some notes about the `Mailboxes.text` file above.

Designating the Postmaster
: Each site has a person who is appointed to be the Postmaster. The Postmaster receives all mail messages that cannot be delivered. In addition, people outside your site communicate about the operation of your mailer with the Postmaster. Note that it is a requirement that any mail server that communicates on the Internet must have a Postmaster.

  In the sample file, the following form appoints Susan to be the Postmaster, and tells the Mail server to place mail addressed to Postmaster in Susan's inbox on the local machine:

```
(define-local Postmaster Susan)
```

Designating the Users Who Receive Mail on this Host
: The `deliver-local` form indicates all people who receive mail on this mail server:

```
;;; People who get mail delivered on White
(deliver-local Robert Susan Peterson Smiley)
```

  The `deliver-local` form causes mail for Robert to be placed in the file `>Robert>Mail.text`, and so on for the other users.

Designating the Users Who Receive Hardcopy Mail
: The `deliver-hardcopy` form indicates all people who receive hardcopy mail on

this mail server:

```
;;;These people get mail in hardcopy only
(deliver-hardcopy Marilyn Steve)
```

The `deliver-hardcopy` form causes mail for Marilyn to be placed in the file `>Mail>Hardcopy>Marilyn.mail`.

Designating Aliases

The `define` form is used to make an alias, or synonym. For example, the following form defines "Sullivan" to be an alias for "Susan":

```
(define Sullivan Susan)
```

The Mailer delivers mail addressed to Sullivan to Susan's mailbox.

Designating Mailing Lists

You create a mailing list by providing a list as the second argument to `define`. The following form defines a mailing list for reporting bugs:

```
(define Bug-Genera
    (list Susan Robert Smiley
            ">Mail>Archive>Bug-Genera.text"))
```

Notice that the last member of the mailing list is a file in the directory `>Mail>Archive>`. This causes all mail sent to the `Bug-LispM` mailing list to be archived to a file.

A Sample Options.Lisp File

Here we show a sample `>Mail>Static>Options.lisp` file. These options are appropriate for one site, but not necessarily for your site. For information on the various options, see the section "Mailer Options".

```
;;; -*- Mode: LISP; Package: MAILER; Base: 10; Syntax: Common-Lisp -*-

;;; Save failed mail in an archived file
(setf failed-mail-reply-file t)

;;; This mailer handles lots of incoming mail from Arpa
(setf number-of-delivery-processes 2)

(setf notify-hosts
  '(Alderaan Quabbin Riverside))

;;; This host does not have Dialnet hardware
(setf enable-slow-delivery-process nil)

;;; Pegasus is slow to process recipients so allow extra time.
(define-host-recipient-processing-time "Pegasus" "15 seconds")
```

Mailer Operations Window

Most operations involving the Mailer can be done by using the Mailer Operations activity. To select the activity, enter:

```
SELECT O
```

Figure ! shows how the Mailer Operations window appears for a Mailer that has not yet been started.

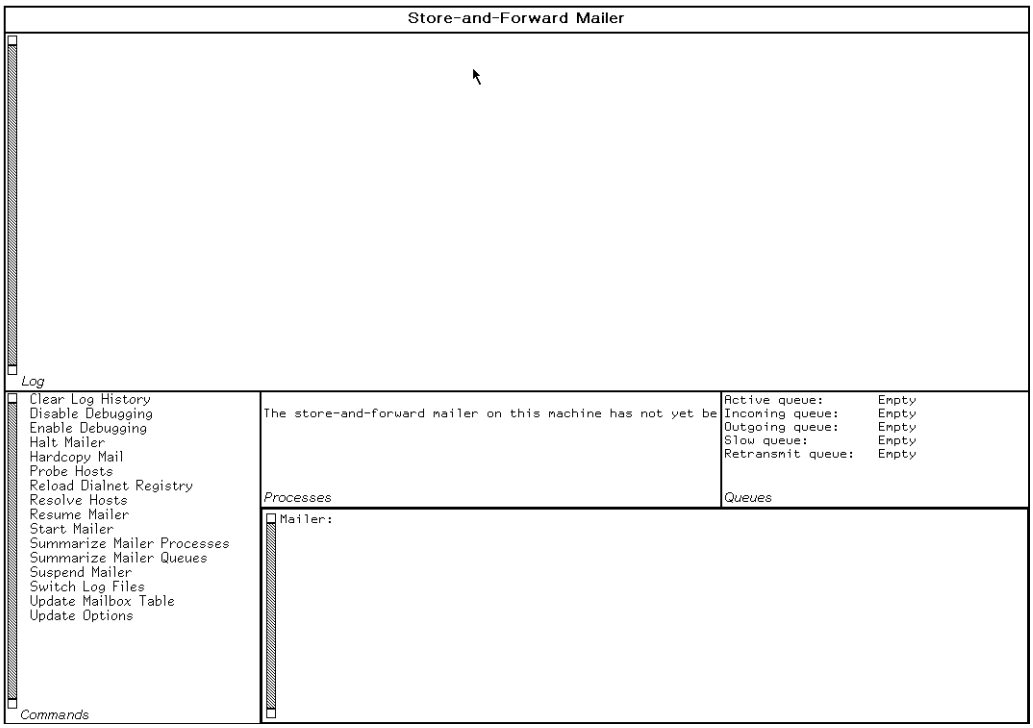


Figure 114. Mailer Operations Window

The window is divided into several panes:

Log — Displays the log file, which is updated dynamically as events are written to the log.

Processes — Displays the status of all Mailer processes.

Queues — Displays the status of all Mailer queues. Incoming contains mail currently being received by the Mailer. Outoing contains mail currently being sent by the Mailer. Slow contains mail that is being sent by a "slow delivery process". Active contains messages that are actually being transmitted by one of the Mailer processes. Retransmit contains messages that are awaiting retransmission and are not in any of the other queues.

Commands Displays the Mailer Operations commands, which you can choose by clicking on them.

Mailer Listener Displays the Mailer Operations commands as you enter them.

For documentation on the commands in the Mailer Operations window, see the section "Mailer Operations Commands".

Mailer Operations Commands

Clear Log History Command

Clear Log History

Clears the Log pane in the Mailer Operations window. This affects the appearance of the window, but does not remove any information from the log file itself.

This command is available only in the Mailer Operations window, which is installed on mailer server hosts.

You can control how much history the log keeps by setting the variable **su::*log-monitor-window-lines***.

su::*log-monitor-window-lines-to-keep* *Variable*

Controls how many lines of logging are kept in the window history. Its value may be a positive integer or **nil**. The default is 500, meaning that the Mailer periodically deletes, from the on-screen history of a log only (not from any associated file), all except the last 500 lines of logging. If it is **nil**, we never trim the log, which can be useful if you often refer to old entries in the on-screen log instead of using the log file, if there is one. Never trimming the log can result in a busy server eventually exhausting its virtual memory, because the history will grow monotonically; however, a server that does little logging will take a long time to do this.

Disable Debugging Command

Disable Debugging

Disables debugging in the Mailer Operations window.

This command is available only in the Mailer Operations window, which is installed on mailer server hosts.

Enable Debugging Command

Enable Debugging *keyword*

Enables debugging within the Mailer Operations window; this causes extra information to be written to the log file.

keyword :Protocol Debugging

:Protocol Debugging
{Yes, No} Whether to include protocol-specific (such as SMTP) debugging information or not. The default is Yes.

This command is available only in the Mailer Operations window, which is installed on Mailer server hosts.

Halt Mailer Command

Halt Mailer

Causes all Mailer processes to finish delivering any mail they have already queued and then to stop themselves. When the Mailer has been halted, all attempts to deliver mail to this host are rejected.

Note that when services are disabled, the Mailer is automatically halted.

The Halt Mailer command in the Mailer Operations window is the same as the function **mailer:halt-mailer**.

This command is available only in the Mailer Operations window, which is installed on Mailer server hosts.

Hardcopy Mail Command

Hardcopy Mail *users-or-all keywords*

Hardcopies the mail for the specified users. The users must be designated as receiving hardcopy mail only; other users can hardcopy their own mail, if desired, by using commands in Zmail.

users-or-all Specifies one or more users or All. All means all users who receive hardcopy mail on this mail server. Those users are designated by a `deliver-hardcopy` form in the `>mail>Static>Mailboxes.text` configuration file.

keywords :Printer, :Quantity

:Printer The printer on which to hardcopy the mail.

:Quantity {All, Last, New} This keyword specifies how much mail to hardcopy. New means all mail that has arrived since the last printing. Last means to print the mail that was just printed; it is useful after a printer failure. All means all mail on file for the user.

This command is available only in the Mailer Operations window, which is installed on Mailer server hosts.

Probe Hosts Command

Probe Hosts *hosts-or-all*

Probes the hosts (which should be mail servers) for which mail is queued. This command causes the Mailer to retry sending the mail to those hosts. The Mailer retries sending queued mail to all hosts at periodic intervals, but this command enables you to retry mail immediately to a host of interest.

For Dialnet hosts, this command also checks to see if the probed host has mail for it, by using the MAIL-PROBE service.

hosts-or-all Specifies which host or hosts should be probed; this is the name of one or more hosts, or All. All means all mail server hosts for which mail is queued.

This command is available only in the Mailer Operations window, which is installed on mailer server hosts.

Resolve Hosts Command

Resolve Hosts *hosts-or-all*

Causes the Mailer to try to resolve the address or other information about a host using the Internet Domain Names facility. The Mailer tries to resolve network information about hosts at periodic intervals, but this command enables you to try to resolve a host of interest immediately.

hosts-or-all Specifies which host or hosts should be probed; this is the name of one or more hosts, or All. All means all mail server hosts whose network information is unresolved.

This command is available only in the Mailer Operations window, which is installed on Mailer server hosts.

Resume Mailer Command

Resume Mailer

Resumes, or "warm boots" the Mailer. The Mailer does not read in its configuration files from disk, but simply tries to resume with its current state.

This command is available only in the Mailer Operations window, which is installed on Mailer server hosts.

Start Mailer Command

Start Mailer

Cold boots (initializes) the Mailer, if the Mailer is not running; otherwise, prints an error message. The Mailer discards all of its important state from virtual memory and restores it from the file system.

Note that when services are enabled, the mailer is automatically started.

This command is available only in the Mailer Operations window, which is installed on Mailer server hosts.

Summarize Mailer Processes Command

Summarize Mailer Processes

Displays a summary of all Mailer processes.

This command is available only in the Mailer Operations window, which is installed on Mailer server hosts.

Summarize Mailer Queues Command

Summarize Mailer Queues

Displays a summary of all Mailer queues.

This command is available only in the Mailer Operations window, which is installed on Mailer server hosts.

Suspend Mailer Command

Suspend Mailer

Suspends the Mailer; stops the Mailer processes when they have reached a consistent state.

This command is available only in the Mailer Operations window, which is installed on Mailer server hosts.

Switch Log Files Command

Switch Log Files

Causes the Mailer to start writing information to a new log file. This happens automatically when the mailer is cold booted by using the Halt Mailer and Start Mailer commands, and when a log file reaches its maximum size. See the variable **mailer:*log-file-maximum-size***.

This command is available only in the Mailer Operations window, which is installed on Mailer server hosts.

Update Mailbox Table Command

Update Mailbox Table

Causes the Mailer to read the file `>Mailer>Static>Mailboxes.text`. If this Mailer distributes forwarding tables, the function also forces new forwarding tables to be written. See the section "Files and Directories Used by the Mailer".

This command is available only in the Mailer Operations window, which is installed on Mailer server hosts.

Update Options Command

Causes the Mailer to reset its options according to the variables set in the `>Mail>Options.lisp` file. Any options not set in that file are reset to their default values.

This command is available only in the Mailer Operations window, which is installed on Mailer server hosts.

Trouble-shooting a Mailer Crash

If a mailer process encounters an error it does not expect, the Mailer is suspended. If all the mailer processes (except the process that is now in the debugger) are running normally, they enter a wait state until the Mailer is warm or cold booted. Incoming mail connections are refused.

When a mailer process is in the Debugger, you will be offered two proceed options specific to the Mailer:

- You can "warm boot" the Mailer, which is the same as using the Resume Mailer command. Warmbooting is faster than cold booting.

- You can "cold boot" the Mailer (reinitialize it from the file system), which is the same as using the Start Mailer command.

We recommend trying to warm boot first. If the error happens again immediately, cold boot the Mailer.

Before rebooting the Mailer, we recommend that you identify the cause of the problem and solve it. Here are the solutions to two typical problem scenarios:

- The Mailer might crash when trying to read one of the `.Mail files` or deliver a message.

This can happen is when there is a disk error in trying to read a file. If it is a disk error, the first thing to do is try to fix it.

If it is not a disk error, or you cannot fix the disk error, rename the offending `.Mail` and `.Work` files to something like *nnnn*`-Mail.bad` and *nnnn*`-Work.bad`. (The file extension should not be `.Mail` or `.Work`, or the Mailer will try again to process the file.) Be sure to rename all versions of the offending files. Then cold boot the Mailer by giving the Start Mailer Command. The `.Mail` and `.Work` files are stored in `>Mail>Dynamic>`.

- Sometimes you want to prevent the delivery of a queued message. To do so, first give the Suspend Mailer command. Then rename or delete the `.Mail` and `.Work` files, then give the Start Mailer command.

Note that you cannot delete or rename a `.Mail` file while the Mailer is running; the Mailer will err. Always give the Suspend Mailer command first.

Mailer Functions

The Mailer is operated by a set of functions in the **mailer:** package. We document these functions for background information only. Instead of calling these functions, you should use the commands in the Mailer Operations Interface. See the section "Mailer Operations Window".

mailer:start-mailer *Function*

Cold boots (initializes) the Mailer, if the Mailer is not running; otherwise, prints an error message. The Mailer discards all of its important state from virtual memory and restores it from the file system.

Note that when services are enabled, the mailer is automatically started.

mailer:restart-mailer *Function*

Warm boots the Mailer, if the Mailer is not running; otherwise, prints an error message. The Mailer assumes that its state in virtual memory is valid and resets and enables the foreground process. The foreground process is responsible for restarting the other processes. You must invoke this function by hand.

mailer:halt-mailer *Function*

Causes all Mailer processes to finish delivering any mail they have already queued and then to stop themselves. When the Mailer has been halted, all attempts to deliver mail to this host are rejected.

Note that when services are disabled, the Mailer is automatically halted.

The Halt Mailer command in the Mailer Operations window is the same as the function **mailer:halt-mailer**.

mailer:update-mailbox-table *Function*

Causes the Mailer to read the file `>Mailer>Static>Mailboxes.text`. If this Mailer distributes forwarding tables, the function also forces new forwarding tables to be written. See the section "Files and Directories Used by the Mailer".

mailer:update-forwarding-tables *Function*

Forces new forwarding tables to be written if this Mailer distributes forwarding tables.

mailer:update-options *Function*

Causes the Mailer to reset its options according to the variables set in the `>Mail>Options.lisp` file. Any options not set in that file are reset to their default values.

mailer:hardcopy-all-mail *Function*

Prints all the saved hardcopy mail, for all users who receive hardcopy mail. Hardcopy mail is discarded after a given number of days (the number is specified as a host option).

mailer:hardcopy-new-mail *Function*

Prints all mail that has arrived since the last printing, for all users who receive hardcopy mail.

mailer:hardcopy-last-mail *Function*

Reprints the mail that was just printed, for all users who receive hardcopy mail. It is useful after a printer failure.

mailer:hardcopy-all-mail-for-user *user* *Function*

Prints all the saved hardcopy mail for the designated user. Hardcopy mail is discarded after the number of days specified as a host option.

mailer:hardcopy-new-mail-for-user *user* *Function*

Prints all mailed that arrived since the last printing, for the designated user.

mailer:hardcopy-last-mail-for-user *user* *Function*

Reprints the mail just printed, for the designated user. It is useful after a printer failure.

net:notify-hosts *hosts* &optional *message* &key *(:error-p) (:report* **t***) (:output-stream)* *Function*

Sends a notification to *hosts* when a mailer error occurs.

Keyword arguments are:

:error-p
Setting this keyword to **t** enables the function to report all errors encountered. Specifying **nil** (default) for this keyword enables the function to ignore all errors encountered.

:report
Setting this keyword to **t** (default) enables the function to report whether it succeeded in delivering the message. Specifying **nil** enables the function to report only failures in delivering messages.

:output-stream
Using this keyword enables you to redirect output to a specific stream.

Installing and Configuring the Mailer

Installing and configuring the Mailer consists of these steps:

1. Deciding which host will be the mail server.
2. Editing the namespace database to include a mail server.
3. Freeing disk space on the mail server.
4. Creating directories for the Mailer.
5. Creating the `Mailboxes.text` configuration file.
6. Creating the `Options.lisp` configuration file.
7. Loading the Mailer software.
8. Saving a world containing the Mailer.

Once the Mailer is installed, you can test it. This section gives detailed instructions on how to install, configure, and test the Mailer. It also describes how to configure a large site to use multiple mail servers.

Mailer Installation Procedure

To install and configure the Mailer, follow these steps:

1. Decide which host will be the mail server.

You must choose a host to act as the mail server. The mail server must also be a file server. The files on this server should be backed up regularly, because its file system is being used.

The mail server will do the following:

- Run a world containing the Mailer software.
- Store and update the Mailer databases.
- Deliver mail to users who wish to receive it on this host.
- Forward mail to users who receive mail on another host.

Each user who receives mail must have an account and a mailbox on a mail server host.

A site can have more than one mail server. For information, see the section "Configuring Large Sites for Multiple Mail Servers".

2. Edit the namespace database to include a mail server.

For each mail server host, edit the host object to contain the following service entries:

```
EXPAND-MAIL-RECIPIENT CHAOS SMTP
MAIL-TO-USER CHAOS CHAOS-MAIL
STORE-AND-FORWARD-MAIL CHAOS CHAOS-MAIL
```

The `MAIL-TO-USER` service enables the host to deliver mail to local users. The `STORE-AND-FORWARD-MAIL` service enables the host to forward mail to non-local users.

If the host has an address on the Dial network, add these service entries, which enables the services to work on the Dial network:

```
MAIL-TO-USER DIAL SMTP
STORE-AND-FORWARD-MAIL DIAL SMTP
```

Since the mail server is also a file server, the host object should already have the following entry (if not, add it):

```
SERVER-MACHINE YES
```

Save the host object when you are finished.

Next, edit the site object for your site. Add the following property pair to the site object:

```
All-MAIL-ADDRESSES-FORWARD YES
```

This causes the mailers at your site to process mail addressed to any of the hosts in the site.

Save the site object when you are finished.

3. Free disk space on the mail server.

You should free up enough disk space in the FEP file system of the mail server to accommodate the following material:

- A world about 10% larger than the world already there.
- The Mailer sources and binaries, which require approximately 200 LMFS blocks.
- The Mailer databases. An empty database requires a minimum of 20 LMFS blocks. The size of a database increases approximately linearly with the number of messages stored by the mail server; an "average" piece of mail occupies about 2 LMFS blocks.

You can use the Dired (m-X) Zmacs command, the Delete File command, or SELECT F (the File System Editor) to delete unused files or excess versions of files.

4. Create directories for the Mailer.

Using the Create Directory command, create the following directories on the mailer host:

```
>Mail>
>Mail>Static>
>Mail>Dynamic>
>Mail>Hardcopy>
```

Some sites find it useful to archive mail sent to certain mailing lists. If you want to archive mail in one directory, you can create this directory now too:

```
>Mail>Archive>
```

5. Create the Mailboxes.text configuration file.

Using the editor, edit the file >Mail>Static>Mailboxes.text on the mail server host.

The Mailboxes.text file contains information which allows the Mailer to know where to deliver mail. This file defines all the local recipients of mail, including individuals and mailing lists.

Here we show a sample >Mail>Statice>Mailboxes.text file, and then describe the contents of the file. This file is appropriate for sites that use only one mail server; for information on sites that use multiple mail servers, see the section "Configuring Large Sites for Multiple Mail Servers".

```
;; -*- Mode: Lisp; Package: Mailer; Syntax: Common-Lisp -*-
```

```
;; This file belongs on White only.

(define-local Postmaster Susan)

;;; People who get mail delivered on White
(deliver-local Robert Susan Peterson Smiley)

;;;These people get mail in hardcopy only
(deliver-hardcopy Marilyn Steve)

;;; Give Susan the alias Sullivan
(define Sullivan Susan)

;;; The basic mailing list for bugs.  Each person listed
;;; gets a copy of each bug report, and a copy goes to the files.
(define Bug-Genera
    (list Susan Robert Smiley
            ">Mail>Archive>Bug-Genera.text"))

;;; Useful to have a mailing list that allows you to reach everyone
(define everybody
    (list Robert Susan Peterson
          Smiley Marilyn Steve Nelson))

(define sports-followers
        (list Smiley Marilyn Robert)
```

These forms are not evaluated in the normal Lisp way. The Mailer evaluates them specially. You should use the syntax as used in this sample file, and should not, for example, replace `(list x y z)` with `'(x y z)`.

Here are some notes about the `Mailboxes.text` file above.

Designating the Postmaster

Each site has a person who is appointed to be the Postmaster. The Postmaster receives all mail messages that cannot be delivered. In addition, people outside your site communicate about the operation of your mailer with the Postmaster. Note that it is a requirement that any mail server that communicates on the Internet must have a Postmaster.

In the sample file, the following form appoints Susan to be the Postmaster, and tells the Mail server to place mail addressed to Postmaster in Susan's inbox on the local machine:

```
(define-local Postmaster Susan)
```

Designating the Users Who Receive Mail on this Host

The `deliver-local` form indicates all people who receive mail on this mail server:

```
;;; People who get mail delivered on White
(deliver-local Robert Susan Peterson Smiley)
```

The `deliver-local` form causes mail for Robert to be placed in the file `>Robert>Mail.text`, and so on for the other users.

Designating the Users Who Receive Hardcopy Mail

The `deliver-hardcopy` form indicates all people who receive hardcopy mail on this mail server:

```
;;;These people get mail in hardcopy only
(deliver-hardcopy Marilyn Steve)
```

The `deliver-hardcopy` form causes mail for Marilyn to be placed in the file `>Mail>Hardcopy>Marilyn.mail`.

Designating Aliases

The `define` form is used to make an alias, or synonym. For example, the following form defines "Sullivan" to be an alias for "Susan":

```
(define Sullivan Susan)
```

The Mailer delivers mail addressed to Sullivan to Susan's mailbox.

Designating Mailing Lists

You create a mailing list by providing a list as the second argument to `define`. The following form defines a mailing list for reporting bugs:

```
(define Bug-Genera
    (list Susan Robert Smiley
            ">Mail>Archive>Bug-Genera.text"))
```

Notice that the last member of the mailing list is a file in the directory `>Mail>Archive>`. This causes all mail sent to the `Bug-LispM` mailing list to be archived to a file.

6. Create the `Options.lisp` Mailer configuration file.

Using the editor, edit the file `>Mail>Static>Options.lisp` on the mail server host.

This file contains options that customize the Mailer's operation. Even if you do not want to set any options, it is necessary that this file exist; it can be empty, in this case. The Mailer reads this file when it is started, or when you use the Update Options command in the Mailer Operations window.

Here we show a sample `>Mail>Static>Options.lisp` file. These options are appropriate for one site, but not necessarily for your site. For information on the various options, see the section "Mailer Options".

```
;;; -*- Mode: LISP; Package: MAILER; Base: 10; Syntax: Common-Lisp -*-

;;; Save failed mail in an archived file
(setf failed-mail-reply-file t)
```

```
;;; This mailer handles lots of incoming mail from Arpa
(setf number-of-delivery-processes 2)

(setf notify-hosts
  '(Alderaan Quabbin Riverside))

;;; This host does not have Dialnet hardware
(setf enable-slow-delivery-process nil)

;;; Pegasus is slow to process recipients so allow extra time.
(define-host-recipient-processing-time "Pegasus" "15 seconds")
```

7. Load the mailer software.

To install the Mailer system, you should create a special world that includes the Mailer. As is true when creating any other world, it is recommended that you do as little as possible to the environment prior to world creation. In this situation, you should do all of the steps listed above, including editing the namespace and creating the Mailer files, and then cold boot the machine. Once you are in the process of building a new world, do not switch windows or do anything that causes an unnecessary allocation of storage.

First, cold boot the host. After this, disable services, log in to the SYS host, and then use the Load System command to load the Mailer system. After using the Load System command, save the world with the Save World command.

The procedure for installing the mailer system is shown in the following example:

```
Command: Halt Machine
Fep Command: boot
    .
    .
    .
Command: Disable Services

Command: Login Lisp-Machine :Init File None

Command: Load System Mailer
    .
    .
    .

Command: Save World (Complete or Incremental) name-of-file-to-save-it-in
```

8. Save a world containing the Mailer.

When you save the world, you may want to use the Save World Incremental command, which allows you to have an incremental world built on the site-customized distribution world containing the mailer. For information about this command: See the section "Using the Incremental Disk Save (IDS) Facility".

You have now configured the newly saved world to be a mail server for the site.

9. Boot the Mailer world.

Boot the new world using the FEP Boot command. When services are enabled to the mail server, the mailer starts automatically. Shortly thereafter, you can press SELECT O to bring up the Mailer Operations window. Then test the Mailer by sending messages to and from various machines at the site.

Testing and Registering the Mailer

The best way to test the Mailer is to attempt to send mail to various people. Send mail to Postmaster at your local site, to make sure that works. Then, you should send mail to Symbolics so we know about your site and are able to communicate with you. Send a message to Customer-Reports at Symbolics.

```
Customer-Reports@Riverside.SCRC.Symbolics.COM
```

The message should contain the following information:

- The name of your site.
- The name of your mail server.
- How you can be contacted (your name and telephone number), in case we receive your mail and cannot respond.

If you are on Dialnet, include this information as well:

- The Dial network address of your mailer machine, for example, 16175371234.

You can watch what the Mailer is doing by pressing SELECT O on the mail server. This brings up the Mailer Operations window, where you can watch the log output.

Configuring Large Sites for Multiple Mail Servers

A large sites that has more than one file server might benefit by using more than one mail server. This distributes the work of storing and forwarding mail over more than one host, and lets mail continue to be sent when one of the mail servers is down.

To help coordinate mail delivery at sites with several mail servers, the store-and-forward Mailer supports *forwarding tables*. This lets you define users, mailing lists, and aliases in one Mailboxes.text file, and depend on the Mailer to distribute that information to other mail servers at your site. This technique reduces the work required to keep several copies of Mailboxes.text files consistent with one another.

To use this technique, you designate one particular mail server to be in charge of forwarding-table maintenance. This host will write the forwarding tables to the file systems of all the other mail servers at the site. This asymmetry is, in a sense, a further customization of the particular mail server that writes the forwarding tables. The customization is usually done by placing a **zl:setf** of the variable **mailer:forwarding-table-hosts** in the Options.lisp file. For example, you would place this form in the Options.lisp file on the host Fearless:

```
(setf mailer:forwarding-table-hosts
   '("manfred" "natasha" "boris"))
```

Here, the hosts Manfred, Natasha, and Boris receive new forwarding tables from Fearless, the host to which this Options.lisp file belongs. The forwarding table for a given host is written in the file >Mail>Dynamic>Forwarding.text on that host's local LMFS. Fearless itself will never have a Forwarding.text file.

If you want to run the identical init file on all the server machines at a site, the following example may be instructive. Here, a SYS host (Fearless) runs the mailer and is responsible for writing out the forwarding tables. The file server's init file, which all file servers use, includes the following lines:

```
(defmacro file-server-only-on (hosts &body body)
  `(when (or ,@(loop for host in hosts
     collect `(send net:*local-host* :pathname-host-namep
                    ,(string host)))) ,@body))

(file-server-only-on (fearless)
  (setq mailer:forwarding-table-hosts
     '("manfred" "natasha" "boris")))
```

The contents of the forwarding table are derived from the Mailboxes.text file on Fearless. The file Mailboxes.text on Fearless contains the names of all the mailing lists for this network. In addition to the usual forms defining mailing lists, the file contains forms like the following:

```
;; What follows is a global table of mail addresses for the network.
;; There is one entry for each host, listing all of the mail addresses to be
;; forwarded to that host.  This is the only
;; place in which this table should be edited.
;; The forwarding tables for all other hosts are generated from this one.

(DELIVER-TO NATASHA

        ;; Individuals
        Andy Bob Charles David Edgar
```

```
;; Lists
ASAS Audio Audiophiles
BBoard Bikers Bleeding-Hearts Bridge
...)
```

This deliver-to form states that these individuals and lists should be delivered to the host Natasha. It is expected that the Mailboxes.text file on Natasha defines where the individuals receive mail (such as in a deliver-local or a deliver-hardcopy form), and Natasha also defines these mailing lists (ASAS, Audio, and so on).

The file Mailboxes.text on Fearless also contains similar deliver-to forms for Boris, Manfred, and all other hosts with store-and-forward Mailers.

When the mailer on Fearless is started (that is, when services are enabled, or when you use the Start Mailer command), or when the Mailer notices that the local Mailboxes.text file has changed and has been stable for at least 10 minutes, it reads in its Mailboxes.text file and then writes out Forwarding.text files on all the other mailer hosts. Those hosts eventually read in the new Forwarding.text files and their own Mailboxes.text files, merge the two sets of definitions, and carry on.

The Forwarding.text file that Fearless generates for Boris includes forms for hosts with store-and-forward mailers, such as Natasha:

```
; Mailbox forwarding table for BORIS.
; Written 6/10/90 15:33:54 by Mail-Server running on FEARLESS.
; From F:>Mail>Static>Mailboxes.text created on 6/10/90 15:25:24.
; This table is automatically generated by a program.  Do not edit it.

(DELIVER-TO NATASHA

        Andy Bob Charles David Edgar
        ASAS Audio Audiophiles
        BBoard Bikers Bleeding-Hearts Bridge)
```

If Boris gets incoming mail for these individuals or lists, the mail is forwarded to Natasha. There is no entry for Boris in this list, since those entries come from the Mailboxes.text file on Boris.

Sites with multiple mail servers can make use of the define-local form. One common use of define-local is to define an individual Postmaster for each host. Thus, each Mailboxes.text file has a form such as:

```
(define-local Postmaster Susan)
```

A define-local form is a combination of a define and a deliver-local. It defines Postmaster to be Susan, and causes mail to Postmaster to be delivered to Susan on the local mail server. This practice contrasts with just using define, which would deliver Postmaster mail to Susan, wherever Susan ordinarily receives mail. One advantage to using define-local for the Postmaster is realized when a mail server

is experiencing problems. If the Postmaster normally receives mail on that host, there will be no way to reach the Postmaster from other hosts. However, if the Postmaster has accounts and mailboxes on each mail server, mail can delivered on Boris if it is sent explicitly to Postmaster@Boris.

The forwarding tables include information derived from `deliver-to` forms, but not information derived from `define-local` forms, `deliver-local` forms, nor `deliver-hardcopy` forms.

Internet Domain Names

Introduction to Internet Domain Names

The Internet Domain Names system is a collection of specifications and procedures which implement the DOMAIN protocol, which is commonly used on the ARPA Internet. The DOMAIN protocol deals extensively with naming. It was created to address several problems.

One major problem that the Domain system addresses is the management of the ever-growing number of hosts on the Internet. When there were only a few hundred hosts, it was reasonable to keep a master file of hosts in a central location to be copied across the network periodically. As more and more hosts were registered, the Internet administrators found that they wanted to separate the hosts into smaller administrative units. Information about these hosts would then be maintained locally. As a result, the Domain system places these hosts in a tree-structured administrative system.

The second major problem that the Domain system attempts to address is the difficulty encountered when sending mail between different networks. Each network has a different naming scheme. These different naming schemes have hindered the interconnection of various networks. The Domain system attempts to allow connections between networks as diverse as Internet, CSnet, BITnet, UUCP, Symbolics Dialnet, and others.

For instance, in the past, mail addresses looked like:

- *user%host*.CSnet@CSnet-Relay.ARPA
- *random-host*!*uninteresting-host*!*host*!*user*@UCBVAX.ARPA
- adi/*user%host*.BITnet@WISCVM.ARPA

When addresses are automatically generated by various mailers, the results can be combined to make long and complex addresses.

If all the hosts involved are using the Domain system, all these mail addresses may be viewed as:

- *user@domain*

Symbolics implements the Domain specification described in several Requests for Comments (RFCs) available from the Network Information Center, SRI International. Symbolics implements the Domain specification on both TCP and Chaosnet. See the section "References to IP/TCP Protocol Specifications".

How the Domain System is Structured

Domains are administrative entities. There are no geographical, topological, or technological constraints on a domain. The hosts in a domain need not have common hardware or software, nor even common protocols. Most of the requirements and limitations on domains are designed to ensure responsible administration.

The Domain system is a tree-structured global namespace that has a few top-level domains. The top-level domains are themselves subdivided into domains. These domains can be further subdivided into yet more domains, and so on.

The administration of a domain requires controlling the assignment of names within that domain and providing access to the names, addresses, and list of valid services to users both inside and outside the domain.

The top-level domains are:

- **GOV** Government
- **EDU** Education
- **COM** Commercial
- **MIL** Military
- **ORG** Organization (an "other" category)
- **NET** Network administrative entities

Temporarily, the top-level domains also include:

- **ARPA** The current ARPA-Internet hosts

Additionally, the English two-letter codes identifying a country according to the International Standards Organization (ISO) Standard for *Codes for the Representation of Names of Countries* (ISO 3166, International Standards Organization, May 1981) can be used as top-level domains.

Sufficiently large companies can qualify for their own top-level domain. As of this writing, no company has attempted to qualify for a top-level domain.

How Domain Names Are Structured

The structure of a domain name is formally prescribed. Domains names are printed with each level of the domain name separated by a period. The Domain system knows nothing about hosts or sites; it deals only with names. The order of appearance in a domain name goes from the most specific to the most encompassing. For example, one of the Symbolics domain names is:

```
SCRC.Symbolics.COM
```

Based on the structure of the name, we can surmise that `SCRC` is the particular domain within Symbolics, and indeed it corresponds to a site in the Symbolics namespace. Continuing, we deduce that Symbolics is the name of the company that has a domain name of `Symbolics.COM`, and that `COM` is the top-level domain for commercial organizations. It is equally possible that `SCRC` is the name of a host. There is no way to tell *from the name* what `SCRC` is.

A host named "Rocky" in the Aerospace department at Whatsamatta University might have the domain name of:

```
Rocky.Aero.Whatsamatta.EDU
```

Looking from the most general to the most specific, this host is a part of the `EDU` domain. More specifically, it is a part of the domain `Whatsamatta.EDU`. Within the domain `Whatsamatta.EDU`, there is another domain—`Aero.Whatsamatta.EDU`. At this point, there is nothing to tell us if `Rocky` is a domain or a host in a domain. We know that `Rocky` is a host, because it is stated above. But you should always be aware of this potential ambiguity when reading domain names.

A domain name is just a name. The naming convention requires that some authoritative entity agree that it will be responsible for providing information about some domain and will guarantee that the information provided will follow the domain conventions. There is nothing implicitly better, worse, different, or otherwise unusual about the number of segments in a domain name.

As a consequence of the above convention, periods are effectively reserved characters. The domain `Whatsamatta.EDU` should not be referred to as `Whats.a.matta.EDU`. The latter is in an entirely different domain. A name must contain either a character, a numeral, a dash, an underscore, or some combination of these elements. Domain implementations are currently required to be case-insensitive.

How Genera Uses Internet Domain Names

This section describes how Genera implements Internet Domain Names, and how they are related to the Namespace system. For related information, see the section "The Domain System and the Namespace System".

How do Symbolics computers find network-related information?

In the Symbolics networking environment hosts must be able to obtain certain types of information about hosts and users of the network. The Namespace system stores that information in its database, and provides it to hosts that request information. A typical site has one designated namespace server.

Along with the namespace database, Symbolics computers support the Internet Domain Names style of requesting and obtaining network-related information.

Symbolics computers look for naming information as follows:

- They first seek it in the namespace database.
- If the information is not found, and the request involves an Internet Domain Name, they seek the information from hosts on the network called Name servers.

What kind of sites benefit from Internet Domain Names?

This facility is useful for:

- Sites with one or more hosts that use IP/TCP and are connected to the ARPA Internet, or any Internet that uses Domain Names.
- Sites that use Dialnet.
- Any site that uses the Internet Domain Names style of addressing.

When is the Internet Domain Names facility used?

The Internet Domain Names facility is integrated with the generic network system's procedure for finding a path to a host. When a network service is requested from a remote host, the generic network system must find a path to that host. For example, when you send an electronic mail message, the "To" field can contain an Internet Domain Names style of name, such as:

```
To: Customer-Reports@STONY-BROOK.SCRC.SYMBOLICS.COM
```

The generic network system must find the network address of the host named `STONY-BROOK.SCRC.SYMBOLICS.COM` in order to send the message.

How does Genera find a host's network address?

The part of Genera that does this is called the *name resolver*. Specifically, it is code that is part of **net:parse-host**, which is used often by the generic network system. The name resolver first consults the namespace database for this kind of information. If the information is not found, and the name is an Internet Domain Names style of name (with periods separating the components), the name resolver uses the Internet Domain Names facility. These steps are described below.

How does the name resolver search the namespace for an Internet Domain style name?

In this case, the name resolver looks for a namespace whose **internet-domain-name** attribute is `SCRC.SYMBOLICS.COM`, and then looks for a host named `STONY-BROOK` in that namespace. If no such namespace is found, or no host is found in such as namespace, the resolver begins to seek the information from Name servers. A name must contain at least one period to be a candidate for this kind of resolution.

How does the name resolver seek the Internet Domain Names information?

The name resolver determines whether it has the requested information stored in a local cache; this would happen if it had already processed a similar request. This step saves the resolver from making an unnecessary search for information. If the information is not found in the local cache, the resolver seeks the information from another host on the network. The resolver makes a request of the central name resolver, if any host at the site provides the **:domain** service. If not, it makes a request of one of several designated hosts on the network known as Name servers.

How does the name resolver know if the site has a central name resolver?

When a host is booted, or the Reset Network command is given, the host looks in the namespace database in the current site for any hosts that provide **:domain** service. If so, the resolver always makes requests of the central name resolver instead of making requests directly of the Name servers. If no host provides **:domain** service, the host makes direct requests of Name servers. The host consults the **root-domain-server-address** attribute of the site object to find out the addresses of the servers for the top-level ("root") domain.

What namespace objects does the name resolver create?

Because so much of the network software depends on objects being present in the namespace, the name resolver was implemented to create a host object for hosts that were not already stored in the namespace, but were located via some Name server. For the host named `STONY-BROOK.SCRC.SYMBOLICS.COM`, a namespace called `DOMAIN` is created, if not already present. A host object named `STONY-BROOK.SCRC.SYMBOLICS.COM` is created in the `DOMAIN` namespace, if it is not already present.

Symbolics Computers as Central Name Resolvers

Name servers are hosts that provide a service to all hosts on the Internet. A *central name resolver* is a host that provides a service to all hosts at a site; that service is described here.

Some sites gain advantages when they designate a single host to perform most of the name resolution for the entire site. Each host at the site contains the name resolver software, but in this configuration that code does not make requests to Name servers on the network. Instead, it makes a request of the central name resolver host. Note that you can configure your site to have multiple hosts designated as central name resolvers.

A central name resolver receives requests from hosts at the site, and processes them by requesting the desired information from Name servers. When information is returned, the central name resolver shares it with the user host, and also stores it in a local cache. Thus, if a second host at the site requests the same information, the central name resolver can return it quickly, without resorting to another network request.

To designate a host as a central name resolver, you should add the following service attribute to its host object:

```
Service: Set: DOMAIN CHAOS DOMAIN
```

If the resolver supports IP/TCP protocols, you should also add the following:

```
Service: Set: DOMAIN TCP DOMAIN
```

Symbolics Computers as Name Servers

The name resolver lets a Symbolics computer go out to the network to request information from Name servers. In addition, Symbolics computers can be Name servers themselves.

When a Symbolics computer is designated as a Name server, it has a responsibility to provide information to other hosts on the network regarding hosts, users, and other network objects within its domain. When it is booted, it loads a file that defines its domain and some other configuration data. Much of the information that the Name server needs resides in the namespace database. The Genera implementation takes advantage of that, and does not require that the Name server duplicate information already stored in the namespace. When the Name server needs information not present in the namespace, it can be stored elsewhere. The file SYS:SITE;LAUNCH-DOMAIN-SERVER.TEXT contains the pathnames of any additional data files.

A computer that is designated to be a Name server *for the ARPA Internet* must support IP/TCP, because it must be capable of communicating with other hosts on the Internet using IP/TCP.

Note that a Symbolics computer can be a Name server even if it is not connected to the ARPA Internet and does not support IP/TCP. For example, a site that supports only Chaosnet protocols could still use Internet Domain Names to name users and hosts. All that is required is that each host on the network is capable of requesting Name resolution and that the designated Name server is capable of storing and providing the information necessary to resolve Internet Domain Names.

It is not necessary that a Symbolics computer acting as a Name server have the **:domain** service attribute in its host object.

Internet Domain Name Namespace Attribute

During installation you specify the Internet Domain Name to be associated with the namespace in which local hosts are registered, by editing the **Internet Domain Name** attribute of the namespace object that represents the local namespace itself. All hosts that are named within that namespace then inherit the Internet Domain Name that is entered in the namespace object.

For example, the SCRC namespace object might have this attribute:

```
Internet Domain Name:  SCRC.Symbolics.COM
```

SCRC|JUNCO is a host named Junco in the SCRC namespace. Junco inherits the Internet Domain Name of its namespace, so its Internet Domain Name is:

```
Junco.SCRC.Symbolics.COM
```

In some cases a host in that namespace is not in the same Internet domain. An individual host can override the Internet domain of its namespace by entering a value in the **Internet Domain Name** attribute of its host object. In this example the host SCRC|GRACKLE has the Internet Domain Name Grackle.MIT.EDU.

```
Internet Domain Name:  Grackle.MIT.EDU
```

The **Internet Domain Name** attribute of the host object is used solely to override the attribute of the namespace object.

The Domain System and the Namespace System

In many ways, the Domain system and the Namespace system attempt to solve the same problems. Both the Namespace system and Domain System attempt to deal with the issue of naming. Both systems deal with a collection of names that refer to a grouping of machines. In the case of the Namespace system, this collection is called the namespace. In the case of the Domain system, we shall refer to this as a domain or a subtree. However, it is important not to draw too close an analogy between the two.

It might appear that both systems map to "administrative entities". Actually, the Domain system returns attributes that are connected to names. The Namespace system goes beyond the Domain system in describing the hosts, users, printers, and networks within an entity known as a Site. There is nothing in the Domain system that is equivalent to a Site. Humans make the connection between a name and an administrative entity like a site; the Domain system software deals *only* with names.

The major difference between the Symbolics Namespace system and the Domain system is that the space containing the set of all Namespace names is flat, whereas the Domain system is organized as a hierarchy. From one perspective, this hierarchy can be viewed as a tree-structured administrative hierarchy.

Any site which has Symbolics computers must use the Namespace system. Communication with other Symbolics machines within a site can occur without use of the Domain system if the Symbolics machines are in the Namespace system. Any site which wants to communicate with other sites must use the Domain system. If there are non-Symbolics machines at your site and you want to communicate with them via IP/TCP, you should run the Domain system. If you are running Dialnet and Genera, you must use the Domain system.

The Domain system and the Namespace system appear to have information which overlaps. In point of fact, you cannot describe information via the Domain system that is also represented in the Namespace system. In other words, the Namespace System is *always* asked first, and it *always* wins any argument about the validity of any piece of data. If a query about a host that is mentioned in both the Namespace system and in the Domain system occurs, the information from the Namespace system will be used.

There is only one way of assuring that the information in the Namespace system and the Domain system don't conflict: by making a Symbolics computer the primary domain resolver for machines that are in the Symbolics namespace at your site. If this is done, the namespace information will be used to complete the domain information. If this is not done, data integrity will be compromised, since you must manually update all host information in both the Namespace system and the Domain system at the same time.

If you have machines that are not part of the Symbolics namespace, you should have a Symbolics machine serve as the piece of the domain tree that corresponds to the Symbolics namespace, and let any other machine deal with other parts of the domain tree. There is no useful way a machine can be a server for only part of a namespace/subtree. Note that nowhere in this discussion have we mentioned "site", only namespace.

You cannot have a partial representation of the hosts in the Namespace system and the remainder in a domain server elsewhere. Partial representation of information in one domain server and the rest in another domain server is also not allowed. Confusion occurs when there is *not* a single authority for a block of names, when one server has one piece of the namespace/subtree and some other server has the rest of the namespace/subtree. *This restriction is not a characteristic of the namespace implementation nor of any domain implementation. Rather, it is a fact common to naming schemes.* If partial information were allowed, it is easy to see that problems would arise as soon as one server's information differed with another's. There *must* be an authoritative server in any naming scheme.

If your organization already has a domain resolver running on another system, you have two options:

- Move the domain resolver to a Symbolics machine.
- Create a new sub-domain containing the Symbolics machines with a Symbolics machine as a domain resolver.

Summary of the Internet Domain Names Facility

Name resolver

This code is used to resolve network names, such as turning a host name into the correct network address for that host. The code is part of **net:parse-host**. If a name (such as a host name) contains periods, it is an Internet Domain Names style of name. In these cases, the name resolver checks to see if the namespace database contains the information. If not, the name resolver makes a network query for the needed information. The name resolver queries a central name resolver if any are designated at the site. If not, it queries one of the Name servers directly. The host can look at the **Root-Domain-Server-Address** attribute of the site object to find out the addresses of the top-level Name servers. If that attribute of the site object is empty, and no central name resolvers

have been designated for the site, then the name resolver cannot resolve the requested name.

Central name resolver

This is a host that the site depends upon to perform name resolution for all Symbolics computers at the site. A central name resolver is designated by having one or two service attributes for **:domain** service in its host object.

```
Service: Set: DOMAIN CHAOS DOMAIN
Service: Set: DOMAIN TCP DOMAIN
```

To resolve a name, a host first checks the namespace database. If the name is not present in the namespace, the host submits a request to the central name resolver, using the **:domain** protocol. The central name resolver checks its local cache to see if it contains the requested information. If not, it makes a request to a designated Name server. The central name resolver decides which Name server to ask by looking at the **Root-Domain-Server-Address** attribute of the site object. If that attribute of the site object is empty, the central name resolver cannot perform the resolution.

Name server

This is any host that provides information on names and addresses to other hosts, using the DOMAIN protocol. Symbolics computers are capable of being Name servers. The server software is a separately loadable system. Note that a central name resolver serves the hosts within the site, but a Name server also serves hosts outside of the site. Sites can configure one Symbolics computer to be both a central name resolver and a Name server.

Symbolics Dialnet

This section gives reference information on Dialnet. For installation instructions, see the section "Installing Dialnet".

Introduction to Dialnet

Symbolics Dialnet is the component of the generic network system that supports the international dial network. The function of Dialnet is to provide a reliable transport medium over possibly unreliable common carrier facilities. The primary uses of this transport medium are mail transfer and remote login. Mail transfer is handled by the Symbolics mail reading and sending program (Zmail) and by the Symbolics Store-and-Forward Mailer. Remote login is handled by the Terminal program.

As of Genera 8.0, Dialnet addresses use the Domain Name System instead of Dialnet Registries. For information about converting existing Dialnet Registries to the

Domain Name System, see the section "Converting From Dialnet Registries to the Domain Name System".

Dialnet and Internet Domain Names

Internet Domain Names are part of a tree-structured naming scheme used by the Internet community for distributed administration of a very large namespace. Dialnet also uses the Internet Domain Names scheme of naming and addressing.

This section discusses how Dialnet uses the Internet Domain Names capability. For a complete discussion of Internet Domain Names, see the section "Internet Domain Names". For installation instructions, see the section "Installing the Internet Domain Names System". Symbolics networks are represented under the second-level domain name Symbolics.COM, and the Symbolics dial namespace is a third-level domain named DialNet.Symbolics.COM. The namespaces of individual Symbolics customer sites are usually represented as fourth-level domains, subdomains of the Symbolics Dial Network. For example, the dial namespace host named CSNY-Young would be in the CSNY.DialNet.Symbolics.COM domain.

Internet Domain Name Namespace Attribute

During installation you specify the Internet Domain Name to be associated with the namespace in which local hosts are registered, by editing the **Internet Domain Name** attribute of the namespace object that represents the local namespace itself. All hosts that are named within that namespace then inherit the Internet Domain Name that is entered in the namespace object.

For example, the SCRC namespace object might have this attribute:

```
Internet Domain Name:  SCRC.Symbolics.COM
```

SCRC|JUNCO is a host named Junco in the SCRC namespace. Junco inherits the Internet Domain Name of its namespace, so its Internet Domain Name is:

```
Junco.SCRC.Symbolics.COM
```

In some cases a host in that namespace is not in the same Internet domain. An individual host can override the Internet domain of its namespace by entering a value in the **Internet Domain Name** attribute of its host object. In this example the host SCRC|GRACKLE has the Internet Domain Name Grackle.MIT.EDU.

```
Internet Domain Name:  Grackle.MIT.EDU
```

The **Internet Domain Name** attribute of the host object is used solely to override the attribute of the namespace object.

Internet Domain Names form the basis of identifying and addressing mail to Dialnet hosts. A basic understanding of Internet Domain Names is essential to the setup and maintenance of Dialnet at your site. It is important to understand the difference between a Dialnet host that is *on the Internet* from one that is not. Dialnet hosts *on the Internet* have direct access to the nationwide Internet or ARPAnet; these hosts (and/or their sites) should have Internet Domain Names as registered

at the SRI Network Information Center. Sites that are not on the Internet, but that use Dialnet, also have Internet Domain Names, but these must be based on the third-level Domain "Dialnet.Symbolics.Com". This distinction must be made to ensure that Dialnet hosts on the Internet are addressed by their true Internet Domain Names and to ensure that all Dialnet hosts not on the Internet fall under a common and useful domain. See the section "Non-Internet Sites: Example Domain Server Launch File".

The Dialnet Subnets File

Addresses for the dial network are complete telephone numbers, including country and area codes. For North American customers, the country code is 1, so a fully specified number looks like a common long distance sequence. Trunk 7348 in the 577 exchange of the 617 area code would be fully specified as 16175777348.

The mailer always identifies Dialnet hosts by their fully-specified addresses, meaning that the address is represented by its country code, area code, exchange, and so forth. Any given Dialnet address has only one fully-specified form, unique worldwide, regardless of the local conventions for how one dials the phone to connect to that address.

It is not generally appropriate to dial a fully specified address; numbers within the same area code do not require the area code, and often require a 1 prefix if it is a toll call. The Subnets file is used to tell the mailer, for a given telephone number, what actual number to dial in order to connect to that number.

There can only be one Dialnet subnets file at any given site, called SYS: SITE; SUBNETS.LISP. This file consists of some number of Lisp forms. Each form is always a list of alternating keywords and values like this:

```
(:subnet "1xxxyyyyyyy>1800zzzzzzz" :dial "1800zzzzzzz" :cost "1")
```

All three keywords must appear, and they must appear in this order. No other keywords are accepted.

The attribute after the **:subnet** keyword specifies a pattern that must match in order to consider the rest of the particular form. If the match succeeds, the actual telephone number that the modem should dial is described by the attribute after the **:dial** keyword, which may contain modem control characters as well as pattern-matching characters and literal digits. Finally, the attribute after the **:cost** keyword specifies how expensive this call is, and is used to select the cheapest way to route the call if more than one of the **:subnet** patterns matches.

:Subnet Keyword in the Dialnet Subnets File

The Mailer may know the Dialnet addresses for a large number of hosts. It is not necessary to specify every possible binary combination of world-wide phone exchanges and their associated prefixes in order for the mailer to know how to dial the phone. Instead, the attribute following the **:subnet** keyword in the subnets file provides a simple pattern matcher that can be used to express both specific and general dialing rules. The name of each subnet on the dial network gives the input

pattern to the pattern-matching system; these patterns are matched against the combined source and destination addresses for the connection, that is, against the local and foreign telephone numbers.

The pattern consists of two sequences of digits and letters. The digits represent the fixed parts of the pattern and the letters represent the variable parts. The two sequences are separated by a > character, indicating that the left-hand part of the pattern is the calling party and the right-hand part of the pattern is the called party. Contiguous occurrences of the same letter represent the same variable. Variable assignment takes place from left to right. If a letter is seen that has no assignment, the variable sub-sequence is tentatively assigned a value of the corresponding sub-sequence of the pattern to be matched. If the variable has an assignment (binding), or if there is a constant digit, it must match the corresponding part of the pattern to be matched.

A specific example clarifies this. Suppose we are calling from `16175777348` to `14155200142`. Given the subnet pattern `1xxxyyyyyyy>1zzzwwwwwww`, we want to match it against `16175771212>14155200142`. `1` is a fixed constant and matches. `x` has no binding so it is tentatively assigned `617`. Likewise `y` is assigned `5777348`, `z` `415`, and `w` `5200142`. The match is successful and the result is these four bindings.

Now suppose instead the subnet pattern was `1xxxyyyyyyy>1xxxzzzzzzz`. The `x` assignment is the same, `617`, as is the `y` assignment. On the second occurrence of `x`, however, it already has a binding, so this must be matched against the input. `617` does not match `415`, so the whole subnet match fails.

The subnet that best represents a particular phone call is the one with the most minimal variable bindings. So, if we were making the call `16175777348>16175777344`, the pattern `1xxxyyyyyyy>1xxxzzzzzzz` would have only three bindings, and so would be better than `1xxxyyyyyyy>1zzzwwwwwww`, which has four.

:Dial Keyword in the Dialnet Subnets File

The attribute following the **:dial** keyword in a subnets file is used if the pattern match in a **:subnet** attribute succeeds. This **:dial** attribute is a sequence of digits, letters, and punctuation. Digits in this attribute are simply dialed literally. Non-digits are more complicated, and may either stand for digits in the number to be dialed, or for modem control characters.

For example, suppose we are calling from `16175777348` to `14155200142`. Given the subnet pattern `1xxxyyyyyyy>1zzzwwwwwww`, we would get the successful matches:

```
xxx      617
yyyyyyy  5777348
zzz      415
wwwwwww  5200142
```

The **:dial** attribute corresponding to this pattern match might be `91zzzwwwwwww`. The non-digits in this attribute will be filled in from the values obtained from the pattern-match of the **:subnet** attribute, meaning that we will actually dial `914155200142`. We would do something like this if the modem went through a PBX that required dialing 9 to get to an ordinary outside telephone line.

Some telephone systems, such as PBX's, may require you to dial a number to get to an outside line, wait for a second dial tone, and then continue dialing. Many modems support this sort of dialing by allowing you to embed punctuation characters in the string of numbers to dial which cause the modem to take some special action.

To allow you to specify this, if you specify characters in the **:dial** attribute that are not matched by the right side of the **:subnet** attribute, those characters will be sent literally to the modem, rather than eliciting an error message. (Before Genera 8.0, unmatched characters in the right side would cause an error message.) For example,

```
(:subnet "1xxxyyyyyyy>1xxxzzzzzzz" :dial "T9,WPzzzzzzz" :cost "0")
```

might tell a Hayes modem that, in order to dial an outside number from a PBX that is in the same area code as the number to be dialed, it must DTMF-dial a 9, wait for a second dial tone, then pulse-dial the rest of the number.

Note that because unmatched non-digits in the **:dial** attribute will be sent directly to the modem instead of causing an error, typographical errors in this attribute are difficult to catch.

:Cost Keyword in the Dialnet Subnets File

The attribute following the **:cost** keyword should be a small integer which somehow reflects the cost of the call in some convenient metric. Typically this is related to how expensive it is to make the call. If more than one pattern matches a particular address, Dialnet uses the match with the lowest cost. This typically comes into play when an 800 number matches some address that is also matched by a "normal" long-distance line. If there is only one way to reach the given number (only one pattern matched), the cost is ignored.

Here is an example of a typical subnets file:

```
;;; -*- Mode: Lisp -*-

(:subnet "1xxxyyyyyyy>1xxxzzzzzzz" :dial "zzzzzzz"     :cost "0")
(:subnet "1xxxyyyyyyy>1zzzwwwwwww" :dial "1zzzwwwwwww" :cost "5")
(:subnet "1212xxxxxxx>1yyyzzzzzzz" :dial "yyyzzzzzzz"  :cost "5")
(:subnet "1617864xxxx>1617774yyyy" :dial "1774yyyy"    :cost "3")
(:subnet "1xxxyyyyyyy>1800zzzzzzz" :dial "1800zzzzzzz" :cost "1")
```

These mean, respectively:

1. When dialing a call within the same area code, just dial the number.

2. When dialing a number outside the local area code, dial a 1, then the area code and number.

3. When dialing from the 212 area code, you do not have to use a 1 prefix for long-distance calls.

4. Within the 617 area code (Massachusetts), you need to dial a 1 to get from Cambridge (864) to East Boston (774).

5. The cost of a wide-area telephone service (WATS) call is less than a normal long distance call. Note that the cost of WATS is still declared higher than a local call; this is to avoid making a WATS call when a local call would do, leaving the WATS trunks available for those who need them.

6. Note that a typical subnets file that may already be suitable for your telephone system is included in SYS: DIALNET; PROTOTYPE-SUBNETS.LISP. (This is not distributed as SYS: SITE; SUBNETS.LISP, which is where the data must eventually be stored, because it may not be correct for your site and the consequences of misdialing can be expensive.)

The map between abstract subnet patterns and actual dialing sequences is maintained by the **subnet** attributes of the namespace object representing the international dial network. (This network is named **dial|dial**.) Each subnet pattern has associated pairs of indicators and values that encode the actual dialing sequence and the relative expense of the phone call.

Using the Terminal Program with the Dial Network

Once you have set up the hardware and namespace information that describes how your host is connected to the dial network, you can use that host to dial up other hosts. This is an excellent test of the hardware and software configuration, even if you don't usually dial up other hosts. And, of course, it can be very useful in its own right, providing access to hosts accessible only via dial-up lines.

Press SELECT T to get to the Terminal program, then type a host name at the Connect to Host: prompt. Host names are of the form dial|dial|16175777348. That represents the host at address 16175777348 on the network named dial|dial, which in turn is the network named dial in the dial namespace. If you need to make the same call frequently, you can add hosts to your own local namespace (not the dial namespace) with addresses on the dial|dial network. In addition to an address attribute, you will probably want to give such a host a service attribute of:

```
Service:  LOGIN DIAL-RAW TTY-LOGIN
```

If this host is in some other domain (this is likely if it is at some other site), you may want to give that host its own Internet Domain Name which corresponds to the Internet Domain Name established by that host's administrator. Use the namespace editor to add a value for the **internet domain name** attribute of the host object for that host.

Telenet PADs have names in the dial namespace such as dial|boston-telenet-pad. Telenet PADs are listed in the subnets file, so to dial up a PAD you must first load the subnets file. See the section "Loading a Dialnet Subnets File".

Installing Dialnet

This section describes how to install Dialnet. Before performing this installation, you should be familiar with the concepts of Dialnet. For background on Dialnet, see the section "Symbolics Dialnet". To install Dialnet, you perform the following steps:

1. Install the Domain Names Server software.

2. Choose a machine to be your Dialnet host.

3. Install the hardware that makes Dialnet possible.

4. Update the local namespace so that the host knows about the Dialnet hardware.

5. Create the subnets file.

This section covers each of these steps, then discusses how to test the Dialnet installation. Finally, a sample Dialnet installation is presented.

Installing the Domain Name Server for Dialnet

This installation step has two parts: loading the domain name server software, and updating the Namespace database.

Loading the Domain Name Server Software

The instructions for loading the domain server software are included elsewhere in the documentation: See the section "Installing the Internet Domain Names System".

The IP Domain Server system uses a launch file to load all necessary files when it starts operating. The entries in the launch file tell the server to load the Dialnet registries. For example, a fictional site ACME, which is not on the Internet, would have the following launch file:

```
;
;        launch file for IP Domain server
;
; type   domain                  source file or host
;
domain   acme.dialnet.symbolics.com
dialnet  dialnet.symbolics.com     sys:site;public-dialnet-registry.lisp
dialnet  dialnet.symbolics.com     sys:site;private-dialnet-registry.lisp
```

There should be a launch file entry for each dialnet registry file used by the Mailer. For more information on the Domain Server launch file: See the section "Installing the Internet Domain Names System".

Updating the Namespace

Dialnet hosts need to have an Internet Domain name associated with them. See the section "Internet Domain Name Namespace Attribute". Most Dialnet sites belong in subdomains of the DialNet.Symbolics.COM domain. Use the Namespace editor to edit the local Namespace object. Enter a value for the **Internet Domain Name** attribute. For example, a site named Acme would have this entry in its namespace object:

```
Internet Domain Name:  Acme.Dialnet.Symbolics.COM
```

Some sites are already using the Internet domain names and have such names assigned to their hosts. Those sites should continue to use the Internet domain names that they are currently using, and enter the domain name of the namespace to the **Internet Domain Name** attribute of the namespace object.

Any host that needs to override the default **Internet Domain Name** stored in the namespace object can enter a different value for the **Internet Domain Name** attribute of their host object. The **Internet Domain Name** attribute of a host object is used solely to override the attribute of the namespace object.

Choosing a Machine to Be Your Dialnet Host

The machine you choose to be your Dialnet host should be one that is on the dial network most of the time. This is important, since this machine receives mail for later distribution to other Dialnet hosts and to other local hosts. If the host is only occasionally connected to the dial network, it cannot do a reliable or speedy job of delivering mail to Dialnet hosts. The Dialnet host must be a Symbolics computer, since the Dialnet software currently only runs on this machine.

We recommend that your Dialnet host be the same machine as your namespace/file/mail server, since that host is supposed to be up most of the time as well. Also, you should choose a machine that has a local file system (LMFS), since sending and receiving mail over Dialnet requires use of the Mailer on your Dialnet host and the Mailer requires a local file system.

Installing the Dialnet Hardware

The first step in connecting your Symbolics computer to the dial network is to configure a Symbolics machine at your site with a modem. Modems may be ordered from Symbolics. For a list of modems supported, please contact Symbolics.

The mechanics of the physical connection of your host to the dial network depend on the model of the processor.

Instructions for Symbolics 3600 Processors

These instructions are for Symbolics 3600 processors only. For instructions for other 3600-family processors, see the section "Dialnet: Instructions for Other Symbolics 3600-Family Processors".

Most Symbolics 3600 processors contain a Vadic 3450 modem mounted inside the processor cabinet. (If yours does not, refer to the instructions for other 3600-family processors.)

Bring both a modular jack and a male EIA connector out to the I/O bulkhead. The modular jack (labeled **MODEM TELCO**) accepts a standard modular plug from the data circuit provided by the telephone company. Connect the male EIA connector (labeled EIA 4) (via a short cable, which you can make yourself or order from Symbolics, see below) to any one of the three female EIA connectors that provide access to the serial lines (labeled EIA 1, EIA 2, and EIA 3). EIA 1 corresponds to serial unit 1, EIA 2 to serial unit 2, and EIA 3 to serial unit 3.

The cable between EIA 4 and EIA 1, EIA 2, or EIA 3 should convey the following signals on the pins given below:

- Pin 2 (TXD ; Transmitted Data)
- Pin 3 (RXD ; Received Data)
- Pin 4 (RTS ; Request To Send)
- Pin 5 (CTS ; Clear To Send)
- Pin 6 (DSR ; Data Set Ready)
- Pin 7 (SG ; Signal Ground)
- Pin 8 (CXR ; Carrier Detect)
- Pin 20 (DTR ; Data Terminal Ready)

Terminate this cable with one male connector and one female connector. If you prefer not to build such a cable yourself, you can order it from Symbolics.

If your 3600 has been upgraded to support audio and phase-encoded video [via UPGR-SY70], then the gender of the serial ports is male. Construct the cable as described above except that both connectors on the cable should be female. Again, if you prefer not to build such a cable yourself, you can order it from Symbolics.

Earlier versions of the FEP proms installed in 3600 processors do not support all the features of the 3600 serial lines. Dialnet requires FEP version 127 or higher. The easiest way to determine what version of FEP proms is installed is to use the Show Herald command with the keyword :detailed.

```
Show Herald :Detailed Yes
```

One line of the resulting display shows the Fep version.

Instructions for Other Symbolics 3600-Family Processors

These instructions are for all Symbolics 3600-family computers other than the 3600 itself.

These computers do not contain an internal Vadic 3450 modem, but instead expect an external Vadic 3451 or CDS-224 modem to be connected to one of the three serial ports.

Each modem has two electrical connections (excluding the power cord). One of the electrical connectors is similar to the connector the telephone company puts on telephones. Plug this into the telephone jack. The other electrical connector has a 25-pin connector which is standard for RS-232 serial lines. Plug this into one of the serial line ports on the back of your Symbolics computer. For example, if you have a Vadic 3451 modem, which has a pre-installed phone line, you simply connect this to the telephone jack. If you have a CDS-224 modem, you first connect the modular jack to the **line in** on the modem and then to the telephone jack.

Bring the serial lines out to the I/O bulkhead and terminate them in male EIA connectors. On the 3640 and 3645 processors, these connectors are labeled SERIAL 1, SERIAL 2, and SERIAL 3. On the 3670 and 3675 processors, these connectors are labeled EIA 1, EIA 2, and EIA 3. On the 3610, 3620, and 3650 processors, these connectors are labeled SERIAL 1 and SERIAL 2.

The cable connecting the modem to the serial port conveys all the signals described for 3600 cabling; only the gender of the serial port connector differs. See the section "Dialnet: Instructions for Symbolics 3600 Processors".

If you prefer not to build this cable yourself, you can order it from Symbolics.

Using a Hayes-1200 Modem with Dialnet

You can use a Hayes-1200 modem with Dialnet. Here is the configuration for the switches on the Hayes-1200:

Switch 1	Up (factory setting is down). This controls whether the modem takes the state of the DTR line into account or not. The factory setting is to ignore the DTR line. However, our software depends upon being able to hang up the phone line by dropping DTR. If this switch is down, outgoing Dialnet calls do not hang up by themselves; the user has to do it manually.
Switch 2	Does not matter (factory setting is up). This controls whether responses are words (default) or numbers. The software handles either.
Switch 3	Down (the factory setting). This controls whether responses are sent at all. The software assumes this is correctly set. If this is incorrectly set, you get an error message saying `"modem not responding"`.
Switch 4	Up (the factory setting). This controls whether the modem echoes the commands. The software assumes this setting; you get an error message saying `"unknown response"` if this is wrong.

Switch 5	Does not matter (factory setting is down). This controls the state of auto-answer on power up; down means off, up means on. You should make sure that the modem's namespace entry corresponds to this setting in the modem, since there is no software which sets auto-answer on or off.
Switch 6	Up (factory setting is down). Down forces carrier detect signal on interface to be always high; up causes it to reflect real carrier detect.
Switch 7	Does not matter (factory setting is up); setting depends upon phone line installation.
Switch 8	Down (the factory setting). This enables recognizing any commands by the modem; if this switch is wrong, you get an error saying `"modem not responding"`.

Using a Hayes-2400 Modem with Dialnet

You can use a Hayes-2400 with Dialnet. Note that you can only use the Hayes-2400 at 1200 baud. You can configure this modem by sending commands to its RS232 input. You have to configure a new Hayes-2400 before using it with Dialnet unless you do not want to receive incoming calls.

You can configure your modem by either using a terminal or using the SERIAL-PSUEDONET technique in the terminal window. For more information, see the section "Using the Terminal Program with Hosts Connected to the Serial Line". Note that it is easier to use an ASCII terminal.

The major point of incompatibility between the factory settings and the settings required by Dialnet concern whether the modem answers the phone when it rings. The factory ships the modem with this capability disabled. You must enable this capability before you can receive incoming calls. Note that you should also set `AUTOANSWER` to `YES` in the Peripheral attribute of the host connecting to the modem.

By default, the modem does not listen to transitions of DTR and cannot be forced to hang up by dropping DTR unless it is told to do so.

For further information, see the table on pages 4-14 and 4-15 of the Hayes Smartmodem 2400 User's Guide for a comparison of the switch settings for a Hayes Smartmodem 1200 and the software configuration of a Hayes Smartmodem 2400.

The following instructions assume the modem contains the settings from the factory. Additionally, they assume that the modem loads user profile 0 when powered on. Send the following to the modem:

Code	*What it Does*
ATSO=1	Answer after the first ring.
AT&D2	Pay attention to DTR and hang up if it drops.
AT&C1&S1	CD and DSR on when appropriate, instead of always.

AT&W	Store Configuration in nonvolatile storage

Note that you must execute the final AT&W so the parameter settings are preserved through power failures.

You cannot configure the modem to continuously use DTMF (tone) dialing. The modem defaults to pulse-dialing. You can set the modem to tone dialing by specifying a T at the first character in the number to dial in the :DIAL field of the :SUBNET form in your SYS:SITE;SUBNETS.LISP file.

Updating the Local Namespace to Know About the Hardware

You must record the physical connection of your machine to the dial network in the Namespace database so that the generic network system can decide how best to establish connections over the dial network. You can represent this connection by adding a **peripheral** attribute to the *local view* of the Dialnet host object.

To edit the Namespace database, use the Edit Namespace Object command, choosing to edit the host object.

The term *local view* is used here in contrast to the dial namespace view of the object. If you are editing a host that has *already* been identified as being in both the local and the dial namespace view namespaces, you are asked to choose (via a small pop-up menu) between the local and the dial namespace view of the object before you actually begin to edit the object. Choose the local view; there are no servers for the dial namespace, so you would have no way to save your changes.

When editing the host object, use the **peripheral** attribute to represent the modem that connects this host to the dial network. There are five relevant indicators for the peripheral attribute: **unit**, **model**, **baud**, **phone-number**, and **autoanswer**.

unit Corresponds to the serial port number on the I/O bulkhead of the machine. The value should be a number between `0` and `3`, inclusive. The serial port for the console is numbered `0`.

model Corresponds to the type of modem attached to this serial port. The value should be one of the following: `va212`, `va3450`, `va3451`, `cds-224`, or Hayes.

baud Should be `1200`.

phone-number Corresponds to the telephone number of the telephone trunk to which this modem is attached. The phone-number associates this modem with a particular dial network address. (The address, in turn, associates this peripheral with a host in the dial namespace.)

autoanswer Corresponds to the ability of this modem to receive incoming calls from other sites. If you wish to receive calls from other sites, this indicator should have a value of `yes`. If it has a value other than `yes`, incoming calls are not answered and communication with other sites can only be initiated by the local

site.

In general, if two sites wish to communicate over the dial network, at least one of the sites needs to enable `autoanswer`.

An example peripheral attribute might be:

```
peripheral: MODEM UNIT 2 MODEL CDS-224 PHONE-NUMBER 16172704170
AUTOANSWER YES
```

Dialnet requires that your local Dialnet host's Namespace object contains the dial address along with the peripheral entry for the modem. Add the dial address attribute in the host object. This is what appears in the Namespace editor:

`Address:` **Pair:** *Network Token*

Click on **Pair:** and type:

`dial|dial` *your-machine's-phone-number*

your-machine's-phone-number should be the same phone number you provided in the peripheral entry for the modem in the same host object. If the phone number in the *Token* field is not the same as the one in the host's peripheral entry, Dialnet will not work.

The dial address for the host is a string representing the fully specified phone number of the host on the international dial network. The dial address specified should not include information used for modem or PBX control, but should specify only the unaltered phone number of the host. For example, the *Address* attribute entry for a fictional Dialnet host in the 617 area code in the United States (whose country code is 1) is:

Address: `dial|dial 16175551212`

Loading a Dialnet Subnets File

All the information in the subnets file can be loaded into the local host with **mailer:verbosely-load-registries**.

```
(mailer:verbosely-load-registries)
```

You might use this function to load the names and addresses of all Telenet PADs, for example.

The **dial:load-dialnet-registry** function also loads the subnets file, but it does not give you any warnings.

```
(dial:load-dialnet-registry)
```

The names of these two functions date from the days of Dialnet registries, which are now obsolete. Their names might change in a future release.

Note that some system programs also load the Dialnet subnets file. The Store-and-Forward Mailer does this when the mailer is started, and the Internet Domain Name Server program does this when that server is started.

A Complete Dialnet Installation for a New, Non-Internet Site

These instructions are simply a set of examples. They exist to give you a starting point for a more customized installation that meets your site's needs. They do not explain the theory of operation of Mailer, the Domain Name Server, or Dialnet, nor do they go into any detail about how to configure them differently from the examples shown here. For more information, please consult the documentation for those systems.

Assumptions for Dialnet Installation At a New, Non-Internet Site

These examples assume you are installing these systems for the first time, and that you are using a CDS-224 modem. The examples assume that the Dialnet host (the host with the mailer and modem) is called Bigboote, and that your site's domain name is Yoyodyne.Symbolics.COM. See the section "Dialnet and Internet Domain Names".

These examples assume, therefore, that the mailer host is Bigboote.Yoyodyne.Symbolics.COM, which does have its own Internet domain name (meaning it doesn't need to use Dialnet.Symbolics.COM). However, we assume that this site is not directly connected to the Internet (but does speak Internet protocols — TCP/IP — internally to other machines within the site).

These examples also assume that you have the IP-TCP system loaded, in order to make it easier to show, all in one place, what typical services should be specified in namespace objects. If you do not run TCP/IP on the Symbolics machines at your site, you can still run a Mailer, a Domain Name Server, and Dialnet at your site, but you should take care not to include any of the services in the examples which mention the TCP or UDP media (the second word in a triple of service, medium, and protocol).

We assume that basic site configuration has been done. This means that hosts are already named, they all have network addresses on whichever networks you use (typically Chaos, Internet, DNA, or SNA).

Finally, to minimize the number of different hosts we talk about, we assume that you will run your Mailer, Domain Name Server, and Namespace Server on the same host, and that this is also the host with the modem that Dialnet will use.

Procedure for Dialnet Installation At a New, Non-Internet Site

You must perform three separate major operations:

1. Build a world containing the appropriate systems

2. Modify the namespace to add some services to hosts

3. Edit a few configuration files

You can do these in any order, including in parallel if you have a separate machine that can be used for building the world while you make edits. If you intend to do these operations in parallel, you should not do the world-build on your namespace server, since that will prevent you from making the necessary changes to items in the namespace while the namespace server is unavailable.

Building a World for Dialnet Installation At a New, Non-Internet Site

You will need to build a world with the Mailer and Domain Name Server systems loaded. (Dialnet and everything else you need are already a part of the world and do not have to be separately loaded.) If you plan to use TCP/IP, make sure the IP-TCP system is also loaded in this world.

This world will run on whichever host will have the modem and run the Mailer; we are presuming here that this is also your Namespace server.

Modifying the Namespace for Dialnet Installation At a New, Non-Internet Site

Remember to save each object after you have finished modifying it.

1. **Internet Domain Name** — The site's Internet domain name is presumed to be Yoyodyne.Symbolics.COM, and the site's Namespace name is Yoyodyne. Therefore, you should do the following:

   ```
   Edit Namespace Object Namespace Yoyodyne
   ```

   Edit the `Internet Domain Name` field to be `Yoyodyne.Symbolics.COM`.

2. **All Mail Addresses Forward** — You probably want to make all mail go to your Mailer host to be delivered. (Certain complicated sites with more than one mailer may want more complex behavior.) The easiest way to ensure that this happens is to turn on All Mail Addresses Forward. See the section "The All Mail Addresses Forward Site Attribute".

   To do this, do the following:

   ```
   Edit Namespace Object Site Yoyodyne
   ```

   Click on the `All Mail Addresses Forward` field, and set it to `Yes`.

3. **Mailer host** — You should do the following:

   ```
   Edit Namespace Object Host Bigboote
   ```

   Click on the empty `Address` field, and add the Dialnet address of the Mailer host. This is the telephone number of the line that the modem is attached to. This example assumes the North American numbering scheme, where the country-code is simply 1; if you are not in North America, you should ensure that your country code precedes the local phone number so that international dialing will work correctly. Thus, you should add the following to the `Address` field:

   ```
   DIAL|DIAL 16172735715
   ```

This is the host at trunk 5715, exchange 273, area code 617, country-code 1, in the DIAL network of the DIAL namespace.

Next, click on the empty Service field (after all the other Service fields), and add the following services by clicking on successive service fields.

```
DOMAIN CHAOS DOMAIN
DOMAIN TCP DOMAIN
EXPAND-MAIL-RECIPIENT CHAOS SMTP
EXPAND-MAIL-RECIPIENT TCP SMTP
MAIL-TO-USER CHAOS CHAOS-MAIL
MAIL-TO-USER DIAL SMTP
MAIL-TO-USER TCP SMTP
STORE-AND-FORWARD-MAIL CHAOS CHAOS-MAIL
STORE-AND-FORWARD-MAIL DIAL SMTP
STORE-AND-FORWARD-MAIL TCP SMTP
```

Finally, in the Peripheral field, click on Modem and fill in as follows:

```
Peripheral: None  Graphics-Tablet  Kanji-Tablet  Modem  Pad
  Sdlc-Interface  Serial-Pseudonet  Sync-Interface  Other
Unit: 1
Baud: 300  600  1200  1800  2000  2400  3600  4800  7200  9600  19200  56000
Model: Hayes  Cds-224  Va212  Va3451  Va3450
Phone-number: 16172735715
Autoanswer: Yes  No
```

Note that, by specifying Autoanswer: Yes, you have stated that incoming calls will be accepted.

Writing Dialnet Configuration Files

There are three general classes of configuration files required:

- Files the mailer needs to run at all
- Files the domain name server needs to run at all
- Files specific to Dialnet

Note that all three types of files have extra information in them if you are running Dialnet. The following examples assume that you are.

General Mailer Configuration for Dialnet

Directories Create the following directories on Bigboote:

```
>Mail>
>Mail>Static>
>Mail>Dynamic>
>Mail>Hardcopy>
```

OPTIONS.LISP

Create the file >Mail>Static>options.lisp and put the following forms in it. These forms cause the mailer to:

- Keep a format file of failed mail
- Send outgoing mail immediately
- Probe Riverside (the central Dialnet mailer at Symbolics) for mail every six hours
- Put logfiles in >Mail>Logs (you should create this directory if you do this)

Note in particular the host Riverside; this host is where you should generally expect to pick up and drop off Dialnet mail if you are a subdomain of Dialnet.Symbolics.COM. Other mail relays exist that may be closer and can hence save long-distance charges in both directions; contact Customer-Reports@Riverside.SCRC.Symbolics.COM for details.

```
;;; -*- Mode: Lisp; Syntax: Common-Lisp; Package: Mailer; Base: 10 -*-

(setf failed-mail-reply-file t)
(setf deferred-delivery-times t)
(setf deferred-receipt-hosts '("Riverside.SCRC.Symbolics.COM"))
(setf deferred-receipt-times "6 hours")
(setf logs-directory (fs:parse-pathname "LOCAL:>Mail>Logs>"))

;;; End of file.
```

MAILBOXES.TEXT

Create >Mail>Static>mailboxes.text and put into it:

```
;;; This file belongs on Bigboote only.

(define-local Postmaster John-Parker)

(deliver-local John-Parker)

;;; End of file.
```

Any Mailer that can accept mail either from the Internet or from Dialnet is required to have a recipient called Postmaster who can receive trouble reports and similar mail. In this case, a user named John-Parker is responsible for such mail.

General Domain Name Server Configuration

Launch file

Create SYS:SITE;LAUNCH-DOMAIN-SERVER.TEXT and include the following:

```
;;;
;;;       This is the boot file for the Domain Name Server.
;;;

;;;       type domain source file or host (no spaces in
;;;       these pathnames, please!)

domain    Yoyodyne.Symbolics.COM

primary  Yoyodyne.Symbolics.COM  SYS:SITE;YOYODYNE-DOMAIN.TEXT
primary  Dialnet.Symbolics.COM  SYS:SITE;PRIVATE-DIALNET-DOMAIN.TEXT
primary  Dialnet.Symbolics.COM  SYS:SITE;PUBLIC-DIALNET-DOMAIN.TEXT
```

If you do not plan to run Dialnet, you may leave out the lines mentioning it in the above file.

Internet domain file

You must perform this step even if you are not directly connected to the Internet and even if you do not run TCP/IP.

Create SYS: SITE; YOYODYNE-DOMAIN.TEXT (the file specified by one of the primary lines in the launch file above) and put into it:

```
;;; Domain server info for Yoyodyne.Symbolics.COM.  This file is
;;; referenced by domain server boot files
;;; (like sys:site;launch-domain-server.text).
;;; Please update the serial number field of the SOA record when
;;; you alter it.  This file is loaded with origin = Yoyodyne.Symbolics.COM
;;;

@       IN      SOA     Bigboote Postmaster.Bigboote (
                                1               ; Serial.
                                86400           ; Refresh.
                                1800            ; Retry.
                                86400           ; Expire.
                                1800)           ; Minimum.
        IN      NS      Bigboote
        IN      MX      10  Bigboote

;;; End of file.
```

Note that this file does not contain any Internet hosts, because this example assumes that Yoyodyne is not directly connected. Any hosts that Bigboote needs to talk to at the site are findable from the namespace.

Dialnet domain files

Assuming you are running Dialnet, you will need to create files to hold domain information about Dialnet hosts.

The first file you'll need is SYS:SITE;PUBLIC-DIALNET-DOMAIN.TEXT, which is shipped on the distribution tape. It contains the entry for Riverside.SCRC.Symbolics.COM, the default Dialnet host for communication between customers and Symbolics, and looks like this:

```
;;; -*- Mode: Text -*-

;;;
;;; →→→ Load origin is Dialnet.Symbolics.COM. ←←←
;;;

;;; Riverside handles all domain names under SCRC.Symbolics.COM.
Riverside.SCRC.Symbolics.COM. DIAL A      "16172704170"
                              DIAL HINFO  SYMBOLICS-3600 LISPM
                              DIAL WKS  ((MAIL-PROBE         MAIL-PROBE)
                                         (MAIL-TO-USER             SMTP)
                                         (EXPAND-MAIL-RECIPIENT    SMTP)
                                         (STORE-AND-FORWARD-MAIL   SMTP))
SCRC.Symbolics.COM.           DIAL MX 5   Riverside.SCRC.Symbolics.COM.

;;; End of file.
```

Note that all domain names in this file end with a dot, meaning that they are all absolute names, not relative names, so the load origin of the file is effectively ignored anyway.

The second file you will need is one you must create. It holds your local host and any other Dialnet hosts hosts you wish to communicate with. Create SYS:SITE;PRIVATE-DIALNET-DOMAIN.TEXT and put into it:

```
;;; -*- Mode: Text -*-

;;;
;;; →→→ Load origin is Dialnet.Symbolics.COM. ←←←
;;;

Bigboote.Yoyodyne.Symbolics.COM. DIAL A      "16172735715"
                                 DIAL HINFO  SYMBOLICS-3600 LISPM
                                 DIAL WKS  ((MAIL-PROBE      MAIL-PROBE)
                                            (MAIL-TO-USER         SMTP)
                                            (EXPAND-MAIL-RECIPIENT  SMTP)
                                            (STORE-AND-FORWARD-MAIL SMTP))
Yoyodyne.Symbolics.COM.          DIAL MX 5 Bigboote.Yoyodyne.Symbolics.COM.

;;; End of file.
```

Suppose Yoyodyne communicates with some other Dialnet site. This foreign site is called Cavaliers, and its mailer host is called Hong-Kong. This site is not on the Internet, so Yoyodyne must use Dialnet to communicate with it even if Yoyodyne were to join the Internet.

Furthermore, the Cavaliers site has no parent domain to sponsor it, and did not apply to the NIC for a domain name, so it must use the Dialnet.Symbolics.COM domain. Hence, the full domain name of the mailer host is Hong-Kong.Cavaliers.Dialnet.Symbolics.COM.

To communicate with this site, a description of it must be added to SYS:SITE;PRIVATE-DIALNET-DOMAIN.TEXT, resulting in a complete file of:

```
;;; -*- Mode: Text -*-

;;;
;;; →→→ Load origin is Dialnet.Symbolics.COM. ←←←
;;;

Bigboote.Yoyodyne.Symbolics.COM. DIAL A      "16172735715"
                                 DIAL HINFO  SYMBOLICS-3600 LISPM
                                 DIAL WKS ((MAIL-PROBE      MAIL-PROBE)
                                           (MAIL-TO-USER         SMTP)
                                           (EXPAND-MAIL-RECIPIENT SMTP)
                                           (STORE-AND-FORWARD-MAIL SMTP))
Yoyodyne.Symbolics.COM.          DIAL MX 5 Bigboote.Yoyodyne.Symbolics.COM.

Hong-Kong.Cavaliers              DIAL A      "85251234567"
                                 DIAL HINFO  SYMBOLICS-3600 LISPM
                                 DIAL WKS ((MAIL-PROBE       MAIL-PROBE)
                                           (MAIL-TO-USER         SMTP)
                                           (EXPAND-MAIL-RECIPIENT  SMTP)
                                           (STORE-AND-FORWARD-MAIL SMTP))
Cavaliers                        DIAL MX 5 Hong-Kong.Cavaliers

;;; End of file.
```

Note several points here:

- There are no dots after "Cavaliers". This is because this file is loaded with origin Dialnet.Symbolics.COM (based on what the launch file says), which means that any name that is not followed by a dot is a relative name to which Dialnet.Symbolics.COM should be appended. This makes the file easier to read if most or all domains in the file are subdomains of the origin domain.

- We are using a country code of 852, city code of 5, and a random phone number of 1234567 to reach this host. The subnets file will instruct the mailer how to dial this phone number on the phone line it is connected to.

- The MX entries cause mail sent to just the site (for example, mail to John-Parker@Yoyodyne.Symbolics.COM or Penny-Priddy@Cavaliers.Dialnet.Symbolics.COM) to be routed to the host with the mailer, so John-Parker's mail will actually be delivered to Bigboote.Yoyodyne.Symbolics.COM and so forth.

Dialnet-Specific Files

Dialnet-specific files

You must create SYS:SITE;SUBNETS.LISP, which describes, given some phone number, what actual digits to dial from your phone system to make the connection. A typical site that doesn't have a PBX and doesn't make international calls may be able to get by with the following information, which can be copied directly from SYS:DIALNET;PROTOTYPE-SUBNETS.LISP:

```
;;; -*- Mode: LISP; Package: DIAL; Base: 10; Syntax: ZetaLisp -*-

;;; Call prefix information.
;;; Common situations; your phone system may operate differently.

(:SUBNET "1xxxyyyyyyy>1zzzwwwwwww" :DIAL "1zzzwwwwwww" :COST "5")
(:SUBNET "1xxxyyyyyyy>1800zzzzzzz" :DIAL "1800zzzzzzz" :COST "1")
(:SUBNET "1xxxyyyyyyy>1xxxzzzzzzz" :DIAL "zzzzzzz"     :COST "0")
```

Testing the Dialnet Installation At a New, Non-Internet Site

After writing these configuration files and building the world, you can boot it on the Mailer host. Make sure you remember to cable the modem.

A simple test of modem connectivity can be made as follows. First, we assume that you have not yet enabled services, so the Mailer has not come up yet. This means that you should evaluate `(mailer:verbosely-load-registries)` in a Lisp Listener, which will load the file SYS: SITE; SUBNETS.TEXT. (The Mailer runs this function as part of its initialization when services are enabled, so you must do this only if you have not enabled services yet.) If you get any warnings from this function besides a message telling you it loaded a file, you either have old Dialnet registries from some pre-Genera 8.0 installation of Dialnet (see the section "Converting From Dialnet Registries to the Domain Name System") or you're missing the subnets file.

Second, press `SELECT T` to get the Terminal Window and type the address of a host that has a modem and is known to be working, such as `DIAL|DIAL|16172704170` (for example, Riverside). The modem should dial the number and you should become connected, assuming that the remote host's phone line is available.

If this test works, the Mailer can at least dial the phone, so you can enable services and start trying to send Dialnet mail.

Modifications to Dialnet Installation for Internet Sites

If you are an Internet site (meaning you can pass IP packets directly to the Internet) and you wish to install Dialnet (presumably to communicate with Dialnet-only sites without using Riverside as a gateway), the installation procedure is almost correct as it stands. See the section "Procedure for Dialnet Installation At a New, Non-Internet Site". The major point of departure is handling the domain files for Internet hosts. It is presumed, if you are an Internet site, that you already have domain files set up; this documentation does not address how to do that.

In particular, let's examine the entry in Riverside's domain file for the aforementioned Bigboote. Riverside's domain file knows this host as:

```
Bigboote.Yoyodyne.Symbolics.COM.   DIAL A      "16172735715"
                                   DIAL HINFO  SYMBOLICS-3600 LISPM
                                   DIAL WKS    ((MAIL-PROBE              MAIL-PROBE)
                                                (MAIL-TO-USER            SMTP)
                                                (EXPAND-MAIL-RECIPIENT   SMTP)
                                                (STORE-AND-FORWARD-MAIL SMTP))
Yoyodyne.Symbolics.COM.            DIAL MX 5   Bigboote.Yoyodyne.Symbolics.COM.
Yoyodyne.Symbolics.COM.            IN   MX 10  Riverside.SCRC.Symbolics.COM.
Bigboote.Yoyodyne.Symbolics.COM.   IN   MX 10  Riverside.SCRC.Symbolics.COM.
```

This entry is substantially similar to the example entry in Bigboote's SYS:SITE;PRIVATE-DIALNET-DOMAIN.TEXT file, as it should be — such information should agree. The changes for Internet use consist of the last two added lines. Here's how they work.

Consider an Internet mailer called Blue-Blaze.Irregulars.EDU which is attempting to deliver mail to Bigboote. Blue-Blaze, not having Bigboote in any of its local information but knowing that it must consult a domain server in the Symbolics.COM domain for information on Yoyodyne, will ask some domain server for Symbolics.COM for Internet mail-exchange (IN MX) RR's, and will receive instructions that all mail for the host Bigboote or its site Yoyodyne should be delivered to Riverside. When Riverside gets the mail, it will notice that Bigboote is really a Dialnet host (since this host has a DIAL address in the domain file but no Internet address), and will act as a relay to deliver the mail via Dialnet to Bigboote.

Thus, the added two lines are the "glue" that tell any Internet-only mailer how to get to the given Dialnet-only mailer by using the cooperating gateway in the middle.

If you want your site to perform the sorts of gatewaying that Riverside does, you should change the last two lines to look like:

```
SomeHost.SomeDomain.           IN   MX 10  YourHost.YourSite.YourDomain.
SomeHost.SomeDomain.           IN   MX 10  YourHost.YourSite.YourDomain.
```

Be careful about trailing dots on the domain names.

It is recommended that, if you intend to speak both Internet and Dialnet protocols, you attempt to route as much of the traffic as possible directly, without involving Riverside as a gateway. This saves use of Riverside for those sites which cannot speak one or the other of these protocols and thus require some third-party gateway to communicate. This means that you should create appropriate entries for sites with which you exchange substantial amounts of mail; for example, if you communicate with SLUG sites extensively, the SLUG Dialnet domain file, modified to use your host as the gateway instead of Riverside, can save you a great deal of editing.

Converting From Dialnet Registries to the Domain Name System

These instructions assume that you already have a running Dialnet installation at your site. If you do not, see the section "A Complete Dialnet Installation for a New, Non-Internet Site".

As of Genera 8.0, Dialnet registries are obsolete. All sites should convert to using the domain name system instead as soon as possible when they install Genera 8.0 on their Dialnet server. Using the Domain Name System (DNS) instead of Dialnet registries makes it possible for Dialnet to work more reliably.

Genera 8.0 provides a conversion tool that will allow you to convert your existing registries to domain name system files. This tool is designed to be used only once. To keep the domain name system files current, you should not continue to maintain Dialnet registries while using the conversion tool. The Mailer warns you if it detects that you still have registries around when it starts up.

To convert, do the following:

1. Load the file SYS:DIALNET;CONVERTING-DIALNET-REGISTRIES-TO-DOMAIN-AND-SUBNET-FILES into your machine.

   ```
   Load File SYS:DIALNET;CONVERTING-DIALNET-REGISTRIES-TO-DOMAIN-AND-SUBNET-FILES
   ```

   This file is not part of the world, since you run this tool only once.

   This file defines the CP command Convert Dialnet Registry. Invoke this command once for each registry file you have (typically, these are the private, user's-group, and public registries). It prompts for the input registry, and also for the two output files. The first output file is a domain name system file containing all the useful information from the original registry. The second file contains any :SUBNET forms found in the registry; The mailer needs these files, but this information is useless to put into the Domain Name System (because subnet information is intended to tell the Mailer how to dial the phone, and this information is different at every site). This second file is not generated if no :SUBNET forms appear in that particular registry.

   The command also takes a keyword argument called :Origin, which overrides the default origin of domains inserted into the output domain name file. Do not specify this option if you don't understand what origins mean in the Domain Name System.

2. Once you have run this command for all registries, you must create a launch file for your domain name server which loads the new domain files. Remove any references in the launch file which start with "Dialnet"; such lines load the Dialnet registries, which are no longer needed.

3. You must also take any subnet files which were generated and combine them into one file called SYS: SITE; SUBNETS.LISP. The mailer will warn you when it starts up if it is expected to use Dialnet and it cannot find this file.

   This program attempts to preserve comments in the original Dialnet registries in the appropriate place in the generated domain files, but you should check its output carefully to ensure that it succeeded, since you will be throwing out the original source of those comments. You might also want to check that there weren't multiple definitions of the same host in different Dialnet registries that produced duplicate entries in the output domain file.

   Note that the domain name file for Riverside is provided for you, in SYS: SITE; PUBLIC-DIALNET-DOMAIN.TEXT. There is also an example subnets file in SYS: DIALNET; PROTOTYPE-SUBNETS.LISP. The contents of this file are not simply distributed in SYS: SITE; SUBNETS.LISP because dialing instructions are different from site to site and the consequences of misdialing can be expensive.

4. Once you have done this, you should throw the old registries away. (You may wish to simply rename them, but ensure that everyone at your site who may modify them knows that they will no longer be read by the Mailer.)

Using the Tape Facilities

Note: All of the tape facilities require that you use a tape drive. The tape drive can exist on the local machine, or you can access a remote tape drive over the network. Make sure that the machine with the tape drive has the **rtape** option in its namespace object. If you want to use a cartridge tape drive, it must be connected to a Symbolics computer.

This section describes different tape facilities. In order to load a tape on a Symbolics machine, you must know in what format the tape was written.

For example, if you receive a tape written in distribution format (such as a source tape from Symbolics), you must use the Restore Distribution Activity to load the tape. Alternatively, if you receive a tape written in FEP-Tape format (such as a world load tape from Symbolics), you must use the FEP-Tape Activity (or the FEP) to load the tape.

Here are the tape facilities that Symbolics provides:

File System Maintenance Program
The File System Maintenance Program can be used for backing up files from a local Lisp Machine file system and reloading those files on the same local Lisp Machine file system, in the same place, at a later date. The intended use of LMFS

backup is to reload files onto the same machine from which they were dumped. For information about this program, see the section "The File System Maintenance Program".

Distribute Systems and Restore Distribution Facilities
The Distribute Systems and Restore Distribution Facilities are used to distribute transportable systems and libraries, defined by system declarations on logical hosts, from one site to another. Use these facilities to transport many files within a system, rather than to transport just a few, unrelated files.

Carry-Tape Program
The Carry-Tape Program provides a means of dumping selected files or sets of files to magnetic tape (cartridge or industry-compatible) and loading them at a later time, possibly at a different site. Using Carry-Tape, you can dump files from any host or set of hosts, and reload them to any place on any host.

FEP-Tape Activity
The FEP-Tape Activity provides a means of writing and reading the cartridge tapes used to distribute world loads and (for 3600-family machines) microcode files.

Table ! describes each tape facilities and its restrictions. For more information about this, see the section "Symbolics Tape Drives and Cartridges".

File System Maintenance Program

Purpose Backing up the file system on Symbolics computers.

Restrictions Use only for LMFS files; may only restore to Symbolics computers.

Tape drive must be local?
No.

Can span multiple tapes?
Yes.

Is tape verifiable? Yes.

Can write more than 20 megabytes per cartridge tape on 3600-family machines?
Yes.

Tapes readable by Lisp or the FEP?
Lisp.

Distribute Systems Activity

Purpose Distributing software which is layered (not world loads).

Restrictions Requires definition of logical pathnames.

Tape drive must be local?
No.

Can span multiple tapes?
If you create multi-tape distributions, each resulting tape is an individually restorable single tape. You must enforce correct ordering when restoring the tapes, though.

Is tape verifiable? No.

Can write more than 20 megabytes per cartridge tape on 3600-family machines?
No

Tapes readable by Lisp or the FEP?
Lisp.

Carry-Tape Program

Purpose Moving files between hosts.

Restrictions More difficult to use when many pathnames are needed.

Tape drive must be local?
No.

Can span multiple tapes?
No.

Is tape verifiable? Yes.

Can write more than 20 megabytes per cartridge tape on 3600-family machines?

Yes.

Tapes readable by Lisp or the FEP?
Lisp.

FEP-Tape Activity

Purpose Writing and reading world load and microcode files.

Restrictions World load or microcode files only.

Tape drive must be local?
No, if reading or writing a tape from Lisp. Yes, if reading a tape from the FEP.

Can span multiple tapes?
Yes.

Is tape verifiable? Yes.

Can write more than 20 megabytes per cartridge tape on 3600-family machines?
Optionally; defaults to write cartridge tapes of less than 20 megabytes.

Tapes readable by Lisp or the FEP?
Lisp and FEP.

Table 7. Tape Facilities and Their Restrictions

Tape Specifications

A tape specification (known as a tape spec) specifies the host and tape drive to use. You can select tape spec parameters, such as the tape density. If you type RETURN, the default will be used. (The default is shown when you are prompted for the tape spec.)

The simplest tape spec is a host name, which specifies the default drive on the host. Use a trailing colon to imply that the preceding token was a host. The first token after a colon is assumed to be a device if not followed by an equal sign (=). Use quotation marks (") to quote strings containing commas, colons, and equal signs.

Here are some sample tape specs:

Host Use the default tape drive on *host*.

*Host:*0 Use tape drive 0 on *host*.

:cart Use the only cartridge tape drive on the default host.

density=1600 Use the default host and tape drive with a density of 1600 bpi.

:1,density=1600 Use tape drive 1 on the default host, with a density of 1600 bpi.

Table ! shows some of the alterable tape specification parameters, with their synonyms.

Parameter	*Synonyms*
Host	Machine
Device	Dev, Unit
Reel	Volume, Vol
Density	Dens, Den

Table 8. Tape Specification Parameters and Synonyms

Symbolics Tape Drives and Cartridges

The 3600 machine model uses a Cipher cartridge tape drive, capable of writing cartridge tapes of only 20 megabytes. The other 3600-family machines use an Archive cartridge tape drive, capable of writing cartridge tapes of more than 20 megabytes. If you write a cartridge tape containing more than 20 megabytes (on any machine model excluding the 3600) you will not be able to read it on a 3600 machine model host. All Ivory-based machines can read cartridge tapes containing up to 38 megabytes.

All Symbolics 3600-family machines use a DC 300 XL/P-type cartridge, which is incompatible with all Symbolics Ivory-based machines. All Ivory-based machines use a DC 2000-type cartridge, which is incompatible with all Symbolics 3600-family machines.

Note: All of the tape facilities require that you use a tape drive. The tape drive can exist on the local machine, or you can access a remote tape drive over the network. Make sure that the machine with the tape drive has the **rtape** option in its namespace object. If you want to use a cartridge tape drive, it must be connected to a Symbolics computer.

Distribute Systems and Restore Distribution Facilities

The Distribute Systems and Restore Distribution Facilities are used to distribute transportable systems and libraries, defined by system declarations on logical hosts, from one site to another. Use these facilities to transport many files within a system, rather than to transport just a few, unrelated files.

To write systems to tape for distribution, use the Distribute Systems command. To restore systems from tape to disk, use the Restore Distribution command. To copy a distribution tape, use the function **dis:copy-distribution-tape**.

Distribute Systems Command

Distribute Systems *systems-and-versions-pairs keywords*

Writes systems to tape for distribution. If you do not specify a system, the Distribute Systems Activity window is selected for you. Distribute Systems lists the systems to write to tape, and asks if you want to perform the Distribute Systems operation. Type `Y` for Yes, `N` for No, `Q` for Quit, or `S` for Selective.

If you choose Selective, each file is listed, and you are asked if you want to distribute that particular file. You can select as many files as you want. After you enter this information, you are prompted for a tape specification, if you did not specify one already.

systems-and-versions-pairs
: A list consisting of items separated by commas, each item being a system name followed by a space and a version number.

keywords
: :Compress Files, :Default Version, :Distribute Patch Sources, :File Types, :Full Length Tapes, :Include Components, :Include Patches, :Included Files Checkpoint, :Machine Types, :Menu, :More Processing, :Output Destination, :Query, :Source Category, :Tape Spec, :Use Cached Checkpoint, :Use Disk

:Compress Files
: {Yes, No} Whether to compress the files when writing them to tape. The default is No. The mentioned default is Yes.

:Default Version
: {Released, Latest, Newest, *version-designator*} Version of the system to distribute if not individually specified in Systems. The default is Released.

:Distribute Patch Sources
: {Yes, No} Whether to include patch sources for system patches. The default is No. The mentioned default is Yes.

:File Types
: {Sources, Binaries, Both, Patches-Only, Default} What file types to distribute. The default leaves it to the specifications in individual **defsystem** forms.

:Full Length Tapes
: {Yes, No} Write all tracks of the tape. Use this *only* if you are sure that you don't have to read the tape on a 3600 Cipher drive. The default is No. The mentioned default is Yes.

:Include Components
: {Yes, No} Whether to include any component systems of the systems being distributed. The default is Yes.

:Include Patches {Yes, No, Selective} Whether to include the patch files for the systems being distributed. The default is Yes. If you include patch files and also distribute source files, the source file corresponding to the patch level, not necessarily the source used for the compilation, is the one included on the tape. For example, suppose you have a system that includes the file blue.lisp.1. You put this file in an editor buffer, modify the code, make a patch file, and then save the buffer with the altered code to blue.lisp.2. When you use Distribute Systems, blue.lisp.2 is distributed, but not blue.lisp.1.

:Included Files Checkpoint
{Patch, Release, None} Limit distributed files to those after this patch number or release name, or None (do not limit). The default is None.

:Machine Types {3600, Imach, All} Specifies whether the systems to distribute should be for 3600-family machines, Ivory-based machines, or all machine types. The default is to distribute systems for all machine types.

:Menu {Yes, No} Whether to use a menu interface to specify details of the distribution. Choosing Yes presents a Distribute Systems frame to select which files are distributed. For detailed information about this frame, see the section "Distribute Systems Activity". The default is No. The mentioned default is Yes.

:More Processing {Default, Yes, No} Controls whether **More** processing at end of page is enabled during the output to interactive streams. The default is Default.

If No, output from this command is not subject to **More** processing.

If Default, output from this command is subject to the prevailing setting of **More** processing for the window. (See the section "FUNCTION M".)

If Yes, output from this command is subject to **More** processing unless it was disabled globally (see the section "FUNCTION M").

:Output Destination
{Buffer, File, Kill Ring, Printer, Stream, Window} Where to redirect the typeout done by this command. The default is the stream ***standard-output***.

:Query {Everything, Yes, Confirm-Only, No} Whether to ask about distributing each file. The default is Confirm-Only. The mentioned default is Everything. Everything queries you about each file being distributed, and again for each system being distributed. Yes queries you about each file being distributed.

Confirm-Only queries you about each system being distributed. No does not query.

For queries about individual files, the possible responses are:

Y Yes, distribute this file.
N No, do not distribute this file.
I Include the remaining files in this system.
B Bypass (do not include) the remaining files in this system.
D Directory. Show the directory containing this file.
E Edit this file.
S Source compare this file.

For queries about systems, the possible responses are:

Y Yes, distribute all files listed in this system.
N No, do not distribute any files listed.
Q Quit. Do not distribute any files listed in this system.
S Selective. Query about each file in this system individually.

:Source Category {Basic, Optional, Restricted, Optional-only, Restricted-only} Indicates which source category or categories to write to tape for distribution. The default is Basic.

:Tape Spec The specification for the tape. The default is the default drive on the local machine. For more information about tape specifications, see the section "Tape Specifications".

:Use Cached Checkpoint
{Yes, No} Use the last checkpoint gathered for this system. Using the cached checkpoint information, if there is any, saves time. But it is safe to use only if you are sure no more patches have been made since the cached information was computed. The default is No. The mentioned default is Yes.

:Use Disk On all machines other than a MacIvory, the choices are {Yes, No}. If Yes, the input is written to disk as a special file that is an image of what would be written to tape. When writing to disk, the distribution plan is not divided into parts according to any size limit. You can use this either to distribute on disk, or when you are preparing a distribution and want to see what files would be written to tape. The default is No. The mentioned default is Yes.

On a MacIvory, the choices are {Tape, Disk, Floppy}. Disk indicates the hard disk, and Floppy indicates the floppy disk. These two values also write a special file that is an image of what would be written to tape. Additionally, when a floppy disk is used, the size limit for a "reel" is set to the capacity of the floppy (that is, 800 Kbytes). Tape means to write to tape; this is the default.

Verify Distribution **Command**

Verify Distribution *keywords*

Reads a Distribution from tape, floppy, or disk, and compares the contents of each file in the distribution to the corresponding file in the file computer. As the files are being compared, a line is typed out for each, to note which file is currently being compared. If there is any discrepancy in length or data, a proceedable error is signalled, describing the failure to compare.

The Verify Distribution command is useful when:

- A new distribution has just been created: the tape, floppy, or disk can be checked quickly against its sources for validity.
- A distribution has just been restored at a client site: the files created can be verified quickly against the distribution on tape, floppy, or disk.

keywords :More Processing, :Output Destination, :Use Disk

:More Processing {Default, Yes, No} Controls whether **More** processing at end of page is enabled during the output to interactive streams. The default is Default.

If No, output from this command is not subject to **More** processing.

If Default, output from this command is subject to the prevailing setting of **More** processing for the window. (See the section "FUNCTION M".)

If Yes, output from this command is subject to **More** processing unless it was disabled globally (see the section "FUNCTION M").

:Output Destination {Buffer, File, Kill Ring, Printer, Stream, Window} Where to redirect the typeout done by this command. The default is the stream ***standard-output***.

:Use Disk On all machines other than a MacIvory, the choices are {Yes, No}. If Yes, the input is read from a tape image file on disk. The default is No. The mentioned default is Yes.

On a MacIvory, the choices are {Tape, Disk, Floppy}. Disk indicates the hard disk, and Floppy indicates the floppy disk. These two values also read a tape image file from disk or floppy. Tape means to read from tape; this is the default.

Restore Distribution Command

Restore Distribution *keywords*

Restores data from tape to disk. This command reads the directory listing of the files on tape, and then restores the files according to the pathname and property information on the tape. For each file it restores, Restore Distribution checks to see if the file already exists in the file system. If it does not exist, it restores the file. If the file does exist, the default behavior is that Restore Distribution states that the file already exists in your file system and skips the file.

keywords :Menu, :More Processing, :Output Destination, :Skip Restoration, :Use Disk

:Menu {Yes, No} Whether to use a menu interface to specify details of the restoration. Choosing Yes presents a Restore Distribution frame to select which files are restored. The default is No. The mentioned default is Yes.

:More Processing {Default, Yes, No} Controls whether **More** processing at end of page is enabled during the output to interactive streams. The default is Default.

If No, output from this command is not subject to **More** processing.

If Default, output from this command is subject to the prevailing setting of **More** processing for the window. (See the section "FUNCTION M".)

If Yes, output from this command is subject to **More** processing unless it was disabled globally (see the section "FUNCTION M").

:Output Destination
{Buffer, File, Kill Ring, Printer, Stream, Window} Where to redirect the typeout done by this command. The default is the stream ***standard-output***.

:Skip Restoration {Yes, No} Whether to skip restoration of files that already exist in the file system. If the file exists in the system and :Skip Restoration is No, Restore Distribution states that the file it was supposed to restore already exists in your file system, and asks where to restore it instead. You supply a pathname. A pathname of Nowhere means skip this file. The default is Yes. The mentioned default is Yes.

:Use Disk On all machines other than a MacIvory, the choices are {Yes, No}. If Yes, the input is read from a tape image file on disk. The default is No. The mentioned default is Yes.

On a MacIvory, the choices are {Tape, Disk, Floppy}. Disk indicates the hard disk, and Floppy indicates the floppy disk. These two values also read a tape image file from disk or floppy. Tape means to read from tape; this is the default.

dis:copy-distribution-tape &key *:use-disk* *Function*

Copies an entire distribution tape. An Accept-Variable-Values menu appears, and you can specify a tape to copy, and a destination. The distribution to be copied can be either a tape or the pathname of a dummy source tape on a disk. The destination must be a tape device.

:use-disk specifies to copy from a dummy distribution file rather an actual tape. The file specified must have been created by using Distribute Systems :use disk Yes.

Restore Distribution Frame

When you use the Restore Distribution command, and specify the keyword :Menu with the value Yes, you invoke the Restore Distribution frame. This frame enables you to examine the list of systems and files on a distribution tape, select the ones you want, and then causes them to be restored to your file system. You can also enter the Restore Distribution frame using the command Select Activity Restore Distribution.

The entire Restore Distribution frame consists of four different sections: the Restore Distribution Command menu, the Files to Restore frame, the Actions during Restore Distribution pane, and the Systems to Restore pane.

Figure ! shows the Restore Distribution frame.

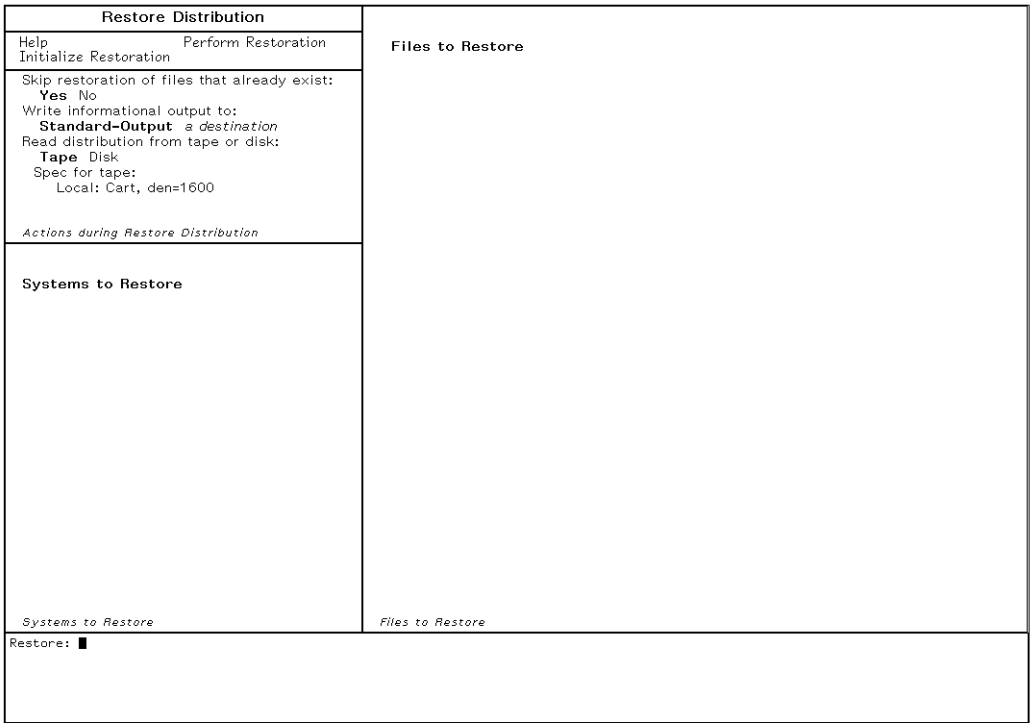


Figure 115. Restore Distribution Frame

Restore Distribution Pane -- the top-left pane

The top-left pane offers offers three commands. To invoke the command, click on the command name.

Help — Displays information about using the Restore Distribution frame.

Initialize Restoration — Reads the directory of all the systems and files on the tape and displays them for your inspection.

Perform Restoration — Restores all files selected in the Files to Restore pane.

Files to Restore Pane -- the right-side pane

The right-side pane displays a list of files, with system header lines.

Actions During Restore Distribution -- the middle-left pane

The middle-left pane enables you to set the parameters for the restoration, such as where to print information about the restoration. You can change one of these parameters by clicking on the value.

The first parameter, "skip restoration of files that already exist", has the following values:

Yes — Skip files that already exist on disk.

No — Notify and prompt for a pathname for each file that already exists on disk.

The second parameter, "write Informational output to", determines (as the distribution is being restored) where the typeout will go. Values are:

```
Standard-Output, or

Destination (Buffer, File, Printer, Stream, or
Window).
```

The default (standard-output) is to write the information on a typeout window on the screen.

The third parameter, "read distribution from device", determines from where the distribution data will be read. Values are machine dependent:

Tape, Disk, or Floppy. — If you choose tape, it prompts you for the tape specification:

```
Spec for tape: Local: Cart, den=1600
```

For more information abouttape specifications, see the section "Tape Specifications".

If you choose disk, the input is written to disk as a special file that is an image of what would be written to tape. You are queried for the path-

name of a tape image file. When disk is chosen, the distribution plan is not divided into parts according to any size limit.

If you choose floppy, you are queried for the pathname of a file on a floppy disk.

Systems to Restore pane -- the bottom-left pane

The bottom-left pane lists the systems on the tape.

- **Initializing a Restore Distribution** — To begin the Restore Distribution process, give the specification for the tape device in the Actions During Restore Distribution pane. Then, click on the command Initialize Restoration to cause the frame to read the directory on the distribution tape.

- **Controlling Selection of Systems and Files** — Initially, Restore Distribution selects all systems and files for restoration. You can control the selection of a file by clicking Left on it to toggle whether it is selected. When a file is deselected, it displays in a smaller font.

  You can control the selection of a system by clicking Left on it in the Systems to Restore pane. When a system is deselected, it is displayed in a smaller font, and its files are completely removed from the file display.

  You can deselect all the files of a system (without deselecting the system itself) by clicking Middle on the header line for that system in the file display. You can select all the files of a system by clicking Left on the header line. This kind of operation is useful, for example, for selecting just a few of a large number of files in a system by first deselecting them all, and then clicking on the few desired files.

  You can deselect all of the systems on the tape by clicking Middle on the title line in the Systems to Restore pane. Select individual systems by clicking Left on the individual systems names. This operation is useful for selecting just a few of a large number of systems.

- **After Selecting The Systems and Files You Want** — When you have selected the correct systems and files (all the files you want to restore are displayed in large letters) click on the command Perform Restoration. The frame checks the distribution tape to make sure it is the same one from which it was initialized. Then it reads the selected files and restores them to the file system.

  While the restoration executes, the informational lines that the operation produces print out on the typeout window over the large pane. If you give some other output destination for the informational output, Restore Distribution writes the output to that place, as well.

- **How to Scroll the Screens** — In the screens with scroll bars, you can scroll the screen display with the SCROLL key, or by pressing m-SCROLL. In addition, you can use m-< to move to the beginning of the display, and m-> to move to the end of the display.

Show Distribution Directory **Command**

Show Distribution Directory *keywords*

Describes the contents of a distribution tape, or a dummy distribution file. For each file in the distribution, this command displays the file's logical pathname, and the physical pathname to which it would be restored. See Figure 101.

keywords :More Processing, :Output Destination, :Use Disk

:More Processing {Default, Yes, No} Controls whether **More** processing at end of page is enabled during the output to interactive streams. The default is Default.

If No, output from this command is not subject to **More** processing.

If Default, output from this command is subject to the prevailing setting of **More** processing for the window. (See the section "FUNCTION M".)

If Yes, output from this command is subject to **More** processing unless it was disabled globally (see the section "FUNCTION M").

:Output Destination {Buffer, File, Kill Ring, Printer, Stream, Window} Where to redirect the typeout done by this command. The default is the stream ***standard-output***.

:Use Disk On all machines other than a MacIvory, the choices are {Yes, No}. If Yes, the input is read from a tape image file on disk. The default is No. The mentioned default is Yes.

On a MacIvory, the choices are {Tape, Disk, Floppy}. Disk indicates the hard disk, and Floppy indicates the floppy disk. These two values also read a tape image file from disk or floppy. Tape means to read from tape; this is the default.

Zmail Commands to Create or Receive ECOs

There are several Zmail commands for handling ECOs. The intent of these commands is to make it possible to distribute patches through electronic mail. The assumption is that you write a distribution to disk, and then using the Mail ECO, encode that distribution in an ascii representation and insert it in a mail buffer. If you have a very large distribution to send, and are afraid that there may be problems with encoding, you can split the process into two stages by encoding first with Make Encoded ECO File.

```
Distribution listing, 12/01/87 18:08:31

Systems on this tape:
  LMFS, TAPE, UTILITIES, SERVER-UTILITIES, HARDCOPY

Files on this tape:                                  Translated pathnames:
 In system LMFS:
    1  SYS:LMFS;LMFS.LISP.105                        Q:>rel-7>sys>lmfs>lmfs.lisp.105
    2  SYS:LMFS;PATCH;LMFS.SYSTEM-DIR.83             Q:>rel-7>sys>lmfs>patch>lmfs.system-dir.83
    3  SYS:LMFS;PATCH;LMFS-94.COMPONENT-DIR.1        Q:>rel-7>sys>lmfs>patch>lmfs-94>lmfs-94.component-dir.1
    4  SYS:LMFS;PATCH;LMFS-94.PATCH-DIR.1            Q:>rel-7>sys>lmfs>patch>lmfs-94>lmfs-94.patch-dir.1
 In system TAPE:
    5  SYS:LMTAPE;TAPPKG.LISP.47                     Q:>rel-7>sys>lmtape>tappkg.lisp.47
    6  SYS:LMTAPE;TAPE.SYSTEM-DIR.127                Q:>rel-7>sys>lmtape>tape.system-dir.127
    7  SYS:LMTAPE;TAPE-74.COMPONENT-DIR.1            Q:>rel-7>sys>lmtape>tape-74>tape-74.component-dir.1
    8  SYS:LMTAPE;TAPE-74.PATCH-DIR.3                Q:>rel-7>sys>lmtape>tape-74>tape-74.patch-dir.3
    9  SYS:LMTAPE;TAPE-74-1.BIN.1                    Q:>rel-7>sys>lmtape>tape-74>tape-74-1.bin.1
 In system UTILITIES:
   10  SYS:SYS;UTILITY-SYSDCL.LISP.51                Q:>rel-7>sys>sys>utility-sysdcl.lisp.51
   11  SYS:PATCH;UTILITIES.SYSTEM-DIR.55             Q:>rel-7>sys>patch>utilities.system-dir.55
   12  SYS:PATCH;UTILITIES-20.COMPONENT-DIR.1        Q:>rel-7>sys>patch>utilities-20>utilities-20.component-dir.1
                  .
                  .
                  .
```

Figure 116. Part of the Display from Show Distribution Directory

In actual fact, you can encode any kind of file as an ECO file, but non-distribution files cannot be loaded with Decode ECO.

Decode ECO (m-X) Zmail Command

Decode ECO (m-X)

Turns an ascii encoded ECO message back into a temporary file and then offers to load that file. You get an error if the file is not a distribution file.

Mail ECO (m-X) Zmail Command

Mail ECO (m-X)

Sends an ECO as a mail message. It prompts for a file containing an encoded ECO. If the file is not yet encoded for mailing, Mail ECO offers to encode it (prompting for a pathname to use to hold the encoded ECO). The encoded ECO is then placed in a message buffer. You can add any comments and terminate the message with END as for any other mail message.

Make Encoded ECO File (m-X) Zmail Command

Make Encoded ECO File (m-X)

Encodes a file for distributing as an ECO. It prompts for a source file and an output pathname to hold the encoded result.

There is also a CP command, Show ECOs, that allows you to see your ECO level:

Show ECOs **Command**

Show ECOs *keywords*

Shows any ECOs (Engineering Change Orders) loaded into your world.

keywords :More Processing, :Output Destination

:More Processing {Default, Yes, No} Controls whether **More** processing at end of page is enabled during the output to interactive streams. The default is Default.

If No, output from this command is not subject to **More** processing.

If Default, output from this command is subject to the prevailing setting of **More** processing for the window. (See the section "FUNCTION M".)

If Yes, output from this command is subject to **More** processing unless it was disabled globally (see the section "FUNCTION M").

:Output Destination {Buffer, File, Kill Ring, Printer, Stream, Window} Where to redirect the typeout done by this command. The default is the stream ***standard-output***.

Carry-Tape Program

The Carry-Tape Program provides a means of dumping selected files or sets of files to magnetic tape (cartridge or industry-compatible) and loading them at a later time, possibly at a different site. Using Carry-Tape, you can dump files from any host or set of hosts, and reload them to any place on any host.

The Carry-Tape Program provides a standard, system-independent interchange medium for exchanging single programs and files between sites. It is meant to fill in a gap between the LMFS backup dumpers and the distribution tape system. It does not require you to prepare files or declarative forms in advance.

The Carry-Tape Program has three components:

- The Carry-Tape Dumper
- The Carry-Tape Loader
- The Carry-Tape Lister

The Carry-Tape Dumper

tape:carry-dump *file-or-files* &key *:device :tape-host :density :reel (:report-stream* **t***) (:query* **t***) :verify :since :author* *Function*

Dumps a file or set of files to a Carry-Tape. You can dump any type of file. Character files are dumped and reloaded using the Genera character set as an interchange medium. Binary files are dumped and reloaded with the proper byte size as long as either of the following is true:

- The file is of one of the system's known canonical types.
- The operating system on which the file resides knows and can supply the byte size.

file-or-files A pathname, file specification, or list of pathnames and/or filespecs. Wildcard pathnames or filespecs may be used. Recursive ("accordion") wildcards may be used to dump subtrees on those hosts that support them. An example of a pathname which has recursive wildcards is:

```
E:>trees>**>*.*.*
```

:device The tape drive device to use. Usually **:cart**.

:tape-host A host object or the name of a host object to use for tape access. **:local** specifies the local tape drive. If you do not specify a host, the dumper uses the standard tape host prompt and defaulting mechanism. If you supply a *tape-host*, you must also supply a *device*.

:density A fixnum, specifying tape density, which may be used when the applicable default is not appropriate.

:reel Can be a string, specifying tape reel name for tape servers that need this information (none of the currently supported ones do).

:report Tells the Carry-Tape Dumper to report its progress as it dumps files. A value of **nil** tells it not to. A value of **t** tells it to report. The default is to report to ***standard-output***. Any value besides **nil** or **t** is expected to be a stream to which the reports will be written.

:query Whether to ask for confirmation about the files to be dumped. The default is **t**.

:verify Whether to compare the tape against the disk after dumping. The default is **nil**.

:since Dump files newer than the date specified.

:author Dump files written by specific user(s).

Currently, Carry-Tape Dumps must fit on one tape.

The Carry-Tape Dumper starts by finding out all available information about the files to be dumped, verifying their existence. It then asks for confirmation, and proceeds to dump all the files specified, without intervention.

Here is an example of using the Carry-Tape Dumper:

```
(tape:carry-dump "swanee:>minerals>*.d*")
To be dumped:
swanee:>minerals>*.d*: 7 files
Is this right? (Y or N) Yes.
Type tape host or spec [default Local :cart]:
Dumping swanee:>minerals>abel.data.3 (5-bit bytes)
Dumping swanee:>minerals>abel.patch-directory.7
.....
Dump complete.
```

For more information about tape specifications, see the section "Tape Specifications".

If you prefer, you can use the equivalent Command Processor command Write Carry Tape instead of **tape:carry-dump**:

Write Carry Tape Command

Write Carry Tape *pathname(s) keywords*

Dumps a file or set of files to a Carry-Tape. See the section "The Carry-Tape Dumper". You can dump any type of file. Character files are dumped and reloaded using the Genera character set as an interchange medium. Binary files are dumped and reloaded with the proper byte size as long as either of the following is true:

- The file is of one of the system's known canonical types.
- The operating system on which the file resides knows and can supply the byte size.

pathname — The pathname(s) of the file(s) to write on the tape.

keywords — :Author, :More Processing, :Output Destination, :Query, :Since, :Tape Spec, :Verify

:Author — Dump files written by specific user(s).

:More Processing — {Default, Yes, No} Controls whether **More** processing at end of page is enabled during the output to interactive streams. The default is Default.

If No, output from this command is not subject to **More** processing.

If Default, output from this command is subject to the prevailing setting of **More** processing for the window. (See the section "FUNCTION M".)

If Yes, output from this command is subject to **More** processing unless it was disabled globally (see the section "FUNCTION M").

:Output Destination
{Buffer, File, Kill Ring, Printer, Stream, Window} Where to redirect the typeout done by this command. The default is the stream ***standard-output***.

:Query — {Yes, No} Whether to ask before dumping files to tape. The default is Yes.

:Since — Dump files newer than the date specified.

:Tape Spec — The host and device for the tape. The default is the default drive on the local machine. For more information, see the section "Tape Specifications".

:Verify — {Yes, No} Whether to compare the tape against the disk after dumping. The default is No.

The Carry-Tape Loader

The Carry-Tape Loader loads files from a carry-tape. The Loader makes no attempt to copy any file properties, including author and creation date. It copies only file contents, and provides reasonable defaults for the target file name.

tape:carry-load &key *:host :density :device :reel (:report* **t***) (:if-exists* **:error***)*
Function

host — A host object or the name of a host object to use for tape access. **:local** specifies the local tape drive. If you do not specify a host, the loader uses the standard tape host prompt and defaulting mechanism. If you supply a *host*, you must also supply a *device*.

density — A fixnum, specifying tape density, which will be used instead of the default.

device — Which tape drive to use. Usually **:cart**.

reel — A string, specifying tape reel name for tape servers that need this information (none of the currently supported ones do).

report — Tells the Carry-Tape Loader to report its progress as it loads files. A value of **nil** tells it not to. A value of **t** tells it to report. The default is to report to ***standard-output***. Any value

besides **nil** or **t** is expected to be a stream, to which the reports will be written. The default is **t**.

if-exists — Whether to error, skip, or supersede files already on the disk. The default is **:error**.

The Carry-Tape Loader begins its operation by reporting the pathnames given to the dumper, and asks if you wish to load all of the files dumped. If only one filespec or pathname was given, it is assumed that you want to load it all, and no question is asked:

```
(tape:carry-load)
Type tape host or spec [default local: cart]: beta
Carry dump made by DCF.
Dump taken at 6/13/88 09:05:22.
Dumped on machine EAGLE.
Dumped: e:>trees>apple.orchard
```

The set of files dumped as a result of each pathname given to the dumper is called a *group*. If many groups were dumped, the loader lists the pathname of each group at the start of its operation, and asks for instructions about which groups are to be loaded (selectively) and which groups are to be skipped:

```
The following groups of files were dumped:
e:>trees>apple.orchard
e:>animals>whales>tails.tales
e:>baseball>runs>foul.*
-----
Load all these files? (ABORT to get out) (Y, Q, or M)
```

The possible responses are:

`Y` Yes — Ignore distinctions of group, and proceed normally, querying about each file.

`Q` Query — Query about each group. The options are:

- `N` No — Don't load the group.
- `Y` Yes — Load the group.
- `P` Proceed — Load this and all succeeding groups.

Proceed as below for those groups that are selected for loading.

`M` Menu — Same as `Q`, but present a multiple-choice menu instead of querying for each group.

If you do not want to load anything, you can press `ABORT` at any time to stop the loader.

The Carry-Tape Loader can either query for the target location of each file to be loaded, or proceed in semi-automatic mode, in which the host and directory from which each file was dumped are used as a key to target loading of subsequent files from that host and directory. The name, type, and version of each file to be loaded are developed automatically from the name, type, and version of the file that was dumped, by means of the same mechanism used by ordinary file copying.

The normal action of the Carry-Tape Loader is to query for each file, with a query of the following form:

```
Load SWANEE:>minerals>rock5.data.6 into BULLWINKLE:/usr2/jones/rock5.data?
        (Y, N, O, H, or A)?
```

The following responses apply:

Y, SPACE	Load the file into the place specified. The host and directory shown remain the default target directory for all files from this host and directory at the site writing the tape.
N	Do not load this file at all. The host and directory shown remain the default target directory for all files from this host and directory at the site writing the tape, in spite of this.
O	Prompt for another place in which to put this file. The host and directory into which this file is then loaded become the default for all subsequent files from the same host and directory at the site writing the tape. You are queried again for each subsequent file if the files are from another directory or host.
A	Load the file into the place specified. All further files from the same host and directory at the site writing the tape are then automatically loaded into the same host and directory as this file without querying you.
H	Prompt for another host to correspond to the host offered.

If you prefer, you can use the equivalent Command Processor command Read Carry Tape:

Read Carry Tape **Command**

Read Carry Tape *keywords*

Loads files from a Carry-Tape. The loader makes no attempt to copy any file properties, including author and creation date. It copies only file contents, and provides reasonable defaults for the target file name. See the section "The Carry-Tape Dumper".

keywords	:If Exists, :More Processing, :Output Destination, :Tape Spec
:If Exists	{Error, Skip, Supersede} Whether to Error, Skip, or Supersede files that are already on disk. The default is Skip.

:More Processing {Default, Yes, No} Controls whether **More** processing at end of page is enabled during the output to interactive streams. The default is Default.

If No, output from this command is not subject to **More** processing.

If Default, output from this command is subject to the prevailing setting of **More** processing for the window. (See the section "FUNCTION M".)

If Yes, output from this command is subject to **More** processing unless it was disabled globally (see the section "FUNCTION M").

:Output Destination {Buffer, File, Kill Ring, Printer, Stream, Window} Where to redirect the typeout done by this command. The default is the stream ***standard-output***.

:Tape Spec The host and device for the tape. For more information, see the section "Tape Specifications".

The Carry-Tape Lister

The Carry-Tape Lister describes what is on a Carry-Tape. Once started, it does not interact in any way.

tape:carry-list &key *:device :tape-host :density :reel (:report* **t***) :verbose* *Function*

device Which tape drive to use. Usually **:cart**.

tape-host A host object or the name of a host object to use for tape access. **:local** specifies the local tape drive. If you do not specify a host, the lister uses the standard tape host prompt and defaulting mechanism. If you supply a *tape-host*, you must also supply a *device*.

density A fixnum, specifying tape density, which may be used when the applicable default is not appropriate.

reel A string, specifying tape reel name for tape servers that need this information (none of the currently supported ones do).

report Tells the Carry-Tape Lister to report its progress as it lists files. A value of **nil** tells it not to. The default is to report to ***standard-output***. Any value besides **nil** or **t** is expected to be a stream, to which the reports will be written. The default is **t**.

verbose Whether to provide detailed information about the properties of each file on the tape.

If you prefer, you can use the equivalent Command Processor command Show Carry Tape instead of **tape:carry-list**:

Show Carry Tape **Command**

Show Carry Tape *keywords*

keywords :More Processing, :Output Destination, :Tape-Spec, :Verify

Describes what is on a Carry-Tape. Once started, it does not interact in any way. See the section "The Carry-Tape Dumper".

:More Processing {Default, Yes, No} Controls whether **More** processing at end of page is enabled during the output to interactive streams. The default is Default.

If No, output from this command is not subject to **More** processing.

If Default, output from this command is subject to the prevailing setting of **More** processing for the window. (See the section "FUNCTION M".)

If Yes, output from this command is subject to **More** processing unless it was disabled globally (see the section "FUNCTION M").

:Output Destination {Buffer, File, Kill Ring, Printer, Stream, Window} Where to redirect the typeout done by this command. The default is the stream ***standard-output***.

:Tape Spec The specifications for the tape. For more information, see the section "Tape Specifications".

:Verify {Yes, No} Whether to verify the tape against the file system. The default is No.

FEP-Tape Activity

The FEP-Tape Activity provides a means of writing and reading the cartridge tapes used to distribute world loads and (for 3600-family machines) microcode files. Tapes written with the FEP-Tape Activity can be read with the FEP commands Load Microcode CART: and Disk Restore, in addition to the FEP-Tape program.

Tapes written with the FEP-Tape Activity can be read by a machine that has an initialized FEP file system and a working set of FEP overlay files (Genera does not need to be running). For more information about the FEP system and overlay files: See the section "Overlay (Flod) Files and the FEP".

On Ivory-based machines, the FEP can use only FEP tapes. On Symbolics 3600-family machines, the FEP can use two types of tapes:

1. Initial File System (IFS) tapes.

2. FEP tapes.

When the FEP reads an IFS tape, it discards all of the data previously stored on the disk. The FEP then creates a new root directory, and empty files to hold world loads and microcode files.

Note: When you restore an IFS tape, you lose all of the data on your disk (including all Lisp Machine File Systems partitions). IFS tapes should not be used without consulting with Symbolics Software Support.

Invoking the FEP-Tape Activity

To invoke the FEP-Tape Activity, type this command to the Command Processor prompt:

```
Select Activity  FEP-Tape
```

This invokes a frame that consists of a command menu, a file display pane, and a listener pane with the prompt:

```
FEP-Tape Command:
```

Figure ! shows the appearance of the FEP-Tape frame.

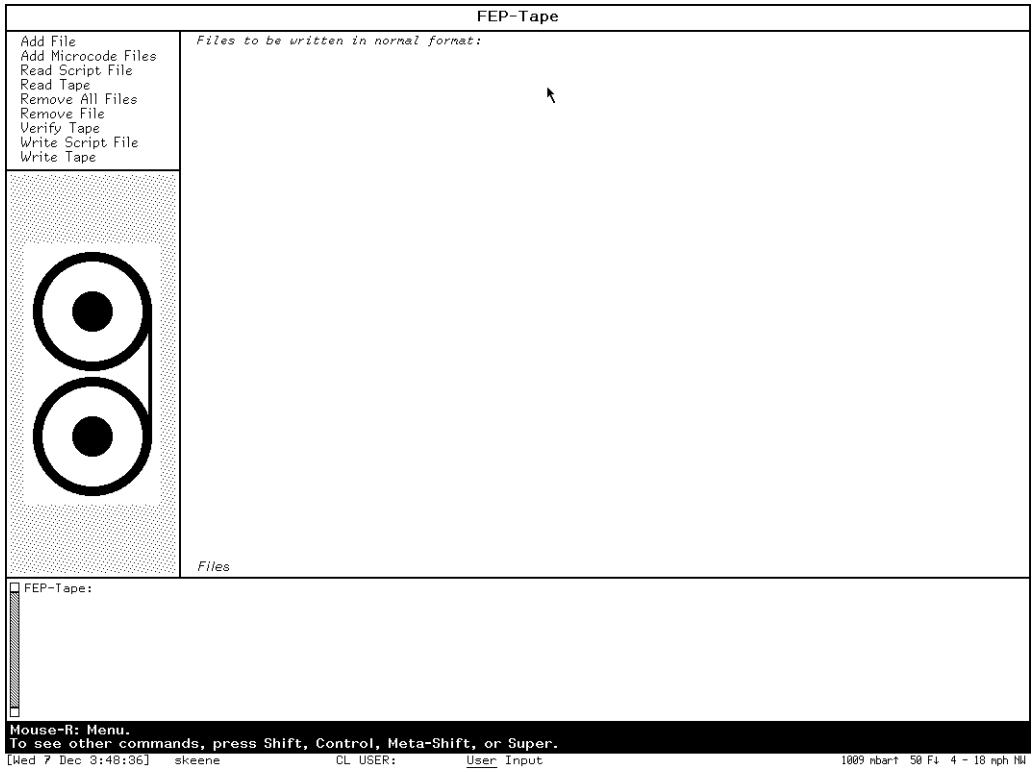


Figure 117. FEP-Tape Window

When preparing to write a tape, the FEP-Tape application lists files to be written to the tape in the file display pane. You can scroll the file display pane with the scroll bar or by using one of the standard scrolling command accelerators (SCROLL, m-SCROLL, m->, m-<, and so on).

The command menu frame lists the FEP-Tape commands. You can type commands in the listener pane at the FEP-Tape Command: prompt, or you can use the mouse to click on commands in the command frame. The following commands are available:

Command	*Definition*
Add File	Adds a single file to the list of files to be written to tape. The command takes arguments that specify: • The pathname. • Whether the file should be written as an initial microcode file, an other file, or both. • A comment to be associated with the file. Only files "other" than the initial microcode files can have comments.
Add Microcode Files	Adds all the microcode files for a particular version of standard microcode to the tape. The command takes arguments to specify the microcode version, whether to write row-major (Genera 7) and column-major (Release 6) array microcode, and whether to set up the initial microcode files.
Read Script File	Reads a script file containing a file set that you have prepared in advance. Use this command to read in a list of files that you have selected to write to tape.
Read Tape	Reads a FEP-Tape tape. This command prompts for the name of each file; you specify where to put the file or if you want to skip reading it all together.
Remove All Files	Removes all of the files listed for writing to tape.
Remove File	Removes one file from the set of files listed for writing to tape.
Verify Tape	Reads a newly written tape set and compares the content of files on disk to the files on the tape.
Write Script File	After specifying your tape set, use this command to save the list of files in a script.
Write Tape	Writes the files to tape. This command takes arguments to specify the host whose tape drive to use (which defaults to local), the unit (which defaults to cart), and whether to write

4-track or 9-track tapes on Symbolics 3600-family machines (which defaults to 4-track).

Writing a Tape Set with the FEP-Tape Activity

1. **Select the set of files to be written on the tape.**

   There are two ways to do this. One way is to read in a script file containing a file set that you have prepared beforehand. You can use the command Read Script File to read the list of files.

   Another way to select the files to be written to tape is to use one or more FEP-Tape commands to specify the files to be written on the tape. You should use the Add File command or the Add Files for Microcode Version command.

   The Add File command adds a single file to the list of files to be written to tape. The command takes arguments that specify the pathname, whether the file should be written as an initial microcode file, an other file, or both, and a comment to be associated with the file. Only files other than initial microcode files can have comments.

   The Add Files for Microcode Version command adds all the microcode files for a particular version of microcode to the tape. The command takes arguments to specify the microcode version, whether to write row-major or column-major array microcode, and whether to set up the initial microcode files.

   If the script file containing a file set has some files you do not want on your FEP tape, use the Remove File or the Remove All Files command to remove some files. The Remove All Files command removes every file listed for writing to tape, and the Remove File command removes just one file from the set of files listed for writing to tape.

2. **Save the list of files in a script file.**

   You can do this with the Write Script File command.

3. **Write the files to tape.**

   Use the Write Tape command to do this. This command takes arguments to specify the host with the cartridge tape drive to use, (default is local) and whether to write 4-track or 9-track tapes. The argument to specify the host is *:host*, and the argument to specify the length is *:Full Length Tapes*.

4. **After writing the files to tape you can verify with the Verify Tape command.**

This command reads a newly written tape set and compares the content of files on disk to the files on the tape.

Note: If you use [Add-File] on a FEP file that is not on the local host, the remote host's name becomes part of the FEP pathname on the tape. This makes the tape unreadable at other sites. To avoid this, always write FEP-tapes on a machine with a local FEP file system that contain any worlds you plan to write to tape. If this machine does not have a tape drive, use remote tape to actually write the tape, but always execute the [Write Tape] command from a machine with the worlds on its local FEP file system.

By default, the FEP-Tape application writes tapes of no more than 19MBytes. This allows the tapes to be read by Cipher tape drives. The FEP-Tape application assumes that the tapes it reads were written for this type of drive.

Some Symbolics 3600-family computers have Archive nine-track cartridge tape drives. Currently, the FEP-Tape application writes or reads 39MBytes using an Archive tape drive only if you give the optional argument :Full Length Tapes Yes to the Write Tape command.

You should only write 39MByte tapes when you are absolutely sure all of the machines that will read the tape have 9-track cartridge tape drives.

For more information, see the section "Symbolics Tape Drives and Cartridges".

Reading a Tape Written with the FEP-Tape Activity

To read a FEP-Tape tape, use the Read Tape command. This scans the tape. For each file, it prompts with the name of the file, and you specify either where to put the file in the file system or that you want to skip reading it all together. This command takes the arguments *:Full Length Tapes* and *:host*.

The FEP command Disk Restore also reads FEP-Tapes.

A FEP-Tape set is a series of one or more cartridge tapes. Ordinarily, a FEP-Tape set contains a world load file and (for 3600-family machines) a set of microcode files.

The FEP cannot create new files. Thus, in order to read a FEP-Tape from the FEP, you must either precreate a suitable set of files while running Lisp, or write over some other existing files.

Printers

Genera supports printers for producing hardcopy. This section provides instructions for installing printers and the printer support software, for configuring your system to recognize printers, and for using the features Genera provides for producing hardcopy.

Genera supports these printers:

- Laser Graphics Printer 2 (LGP2) - A tabletop laser printer supporting the PostScript page description language. You can use this printer for producing text and graphics.

- Laser Graphics Printer 3 (LGP3) - A tabletop laser printer supporting the PostScript page description language. You can use this printer for producing text and graphics.

- Dot-Matrix Printer 1 (DMP1) - An impact dot-matrix printer based on the Fujitsu DPL24 printer.

Symbolics provides limited support for ASCII printers and Serial Dot-Matrix printers. Additionally, Symbolics supports remote PostScript and ASCII printers spooled by UNIX hosts supporting the lpd protocol. To use a PostScript printer, whether from a 3600-family machine, an Ivory, or UNIX, you must have the Symbolics LGP2/LGP3 printer software. Other than printers using lpd protocol, Genera requires that you connect printers using the RS-232 Serial Interface.

Genera enables you to produce hardcopy through the Hardcopy system. The Hardcopy system consists of Hardcopy commands enabling you to produce hardcopy from Lisp, Document Examiner, Zmacs, and Zmail. Additionally, the Hardcopy system contains a programmatic interface enabling you to write programs for producing hardcopy.

Genera provides a Print Spooler for use with shared printers. The Print Spooler automatically manages (queues) multiple hardcopy requests for one or more printers. For more information concerning the Print Spooler, see the section "Managing the Print Queue".

Installing a Printer

In order to install a printer, you have to:

- Uncrate the printer

- Cable the printer to a Symbolics system

- Specify the switch settings

- Load the hardcopy and printer support tape

- Register the printer

- Define the Print Spooler

- Specify Whether a Banner Page Prints (Optional)

Uncrating the Printer

Check the outside of the box containing the printer for unpacking and installation instructions. Please read the instructions carefully before unpacking the printer**.** LGP3 printers are shipped with additional information for installation. For more information, see the section "Special Note for LGP2 and LGP3 Owners". For information concerning temperature, space, ventilation, or electrical requirements (or any other topic relating to unpacking and setup), please call Symbolics Customer Service.

Note that printers are heavy pieces of equipment, and that you should obtain help when lifting the printer out of the box.

Cabling the Printer

Correct printer operation requires proper cabling from your Symbolics computer to the printer. You can either make your own cables, or buy them from Symbolics. Before you can determine the particular cable type you need, you have to determine the printer and interface you are using.

Serial I/O Ports

The Symbolics 3600 and XL400 computers support multiple bulkhead serial ports and one console port. You can achieve the best printer performance using the bulkhead serial ports. Note that using the console port causes the processing of serial information along with other console information, which decreases performance substantially. If you are using a MacIvory or a UX-family machine, you can use the serial ports provided by these machines as well.

Connecting the Printer to your Computer

When connecting a printer to your Symbolics computer through the serial port, make sure the RS-232 connection is correct. Both the printer and the bulkhead serial ports on the computer are considered DTE (Data Terminal Equipment), making the use of a "null-modem" cable necessary. The null-modem connector, also known as a modem eliminator cable, interchanges several signals enabling bidirectional communication. Note that the console serial ports have a null-modem built in.

After you attach the cable, be sure the cable transmits the DTR (Data Terminal Ready) and DCD (Data Carrier Detect) signals to the computer. The computer uses the DTR signal as confirmation that the printer is ready to accept data for printing. The printer support software uses the DCD signal as confirmation that the printer is physically attached to the computer by a serial cable. For more information: See the section "Hardware Description for Serial I/O".

Note that modem signals are not correctly reported after rebooting 3620s and 3650s due to the inability of these machines to query the console about the current state of the modem lines. On these machines, press `LOCAL ABORT` after rebooting.

When connecting the serial cable to a Symbolics computer or console, note the serial port you are using. When registering the printer, register the serial port by

unit number. The console serial port corresponds to unit zero, and the bulkhead serial port corresponds to unit one. Symbolics computers supporting more than one bulkhead serial port display the unit numbers for each port on the bulkhead.

Note: See the Owner's Manual for your printer for determining which printer port accepts an RS232 cable.

For more information on serial hardware interfaces and RS-232 cable specifications, see the section "Hardware Description for Serial I/O". For more information on serial communications and software interfaces for using serial streams, see the section "The Serial I/O Facility".

Specifying the Switch Settings

After you unpack and cable your printer, you must adjust the switch settings for proper communication with your computer. These switch settings include the baud rate, parity setting, and other serial communication parameters. You have to specify these switch settings when registering the printer. If you customize the communication parameters for your printer, you must specify the appropriate values when registering the printer. For more information on specifying communication parameters, see the Owner's Manual for your printer. For more information on serial communication parameters used by Genera, see the section "Parameters for Serial I/O".

Specifying Switch Settings for the LGP2 Printer

You can specify the switch settings for the LGP2 printer following these steps:

1. Plug the printer into an appropriate outlet.

2. Set the mode switch on the back of the printer to 9600 baud (recommended setting for printing from Genera). The LGP2 offers additional communication options, including running at 1200 baud.

3. Turn the printer on using the switch on the left rear side of the printer. A test prints within two minutes, after which you can use the printer.

For more information on setting up an LGP2 printer, see the section "Special Note for LGP2 and LGP3 Owners".

Specifying Switch Settings for the LGP3 Printer

1. Plug the printer into an appropriate outlet.

2. Set the mode switch on the back of the printer to 9600 baud. From the left, switch one is up (off) and switch two is down (on). 9600 baud is the recommended setting for printing from Genera. The LGP3 offers additional communication options, including running at 1200 baud.

3. Turn the printer on using the switch on the left rear side of the printer. A test prints within two minutes, after which you can use the printer. Note that the test page includes several boxes containing graphics. The text beside the second box from the top of the page says "RS-232 9600 Baud."

For more information on setting up an LGP3 printer, see the section "Special Note for LGP2 and LGP3 Owners".

Specifying Switch Settings for an ASCII Printer

Consult the documentation accompanying your ASCII printer for information concerning switch settings. Symbolics recommends using 9600 baud as the serial communication speed and using XON-XOFF handshaking. Note that you must specify the serial communication settings when registering the printer.

Specifying Switch Settings for the DMP1 Printer

The printer support software for DMP1 printers requires that you install a downloadable font cartridge into your printer. The font cartridge stores fonts not built into the printer, but sent from the computer. Note that if you do not see the font cartridge, check the box in which the printer was delivered. The font cartridge is shipped in a small plastic box.

You can install the DMP1 printer following these steps:

1. Install the font cartridge by lifting the plate on the front left side of the printer and inserting the cartridge into the slot with the cartridge name upright.

2. Lift the plate covering the switches on the front left side of the printer.

3. Adjust the switches as follows:

8-position DIP:	On On On On Off Off Off Off
10-position rotary switches:	set both to 6
8-position DIP:	Off On On Off On Off Off On
8-position DIP:	On On Off Off Off Off Off Off
6-position DIP:	Off Off Off Off Off Off

4. Turn on the printer using the red switch on the front of the printer. The carriage moves back and forth several times, after which you can use the printer.

Loading the Hardcopy and Printer Support Tape

If you are installing the first LGP2, LGP3, or DMP1 printer at your site, you must load the hardcopy and printer support tape for that printer.

Depending on the type of printer you are installing, load one of these tapes:

- *Symbolics Laser Graphics Printer Software Interface* tape
- *Symbolics Dot Matrix Printer Sofrtware Interface* tape

You can load the files from tape following these steps:

1. Insert the tape into the cartridge tape drive.
2. Restore the files from tape by specifying the following command:

```
Restore Distribution
```

3. Specify the appropriate tape drive as follows:

```
Hostname: tape-device
```

The files load into the appropriate directories; if a directory does not exist, the computer creates it automatically. The printer support software supports all printers of the type you specify connected to Symbolics computers at your site. For more information on restoring a distribution tape, see the section "Restore Distribution Command".

Registering a Printer

You must register all printers at you site in the namespace database in order to inform the computers at your site that they are available for use. There are three parts to this procedure:

1. Creating a namespace object representing the printer.
2. Associating the printer with its controlling host.
3. Editing the site object.

The procedure for registering a printer requires that you use the Namespace Editor. For more information, see the section "Using the Namespace Editor". Additionally, see the section "Concepts of the Namespace System".

Creating a Namespace Object Representing the Printer

You create printer objects using the namespace editor. The details for using the namespace editor vary depending on the printer and interface, but the following steps are common in all cases:

- Creating a "skeleton" namespace object and filling in those fields independent of printer type (such as the name you choose for your printer).
- Creating an appropriate printer object by filling in all fields particular to the printer and interface.

Creating a Skeleton Namespace Object for Your Printer

You can create a skeleton namespace object for your printer by following these steps:

1. Enter the Namespace Editor by specifying the following command from a Lisp listener:

   ```
   Edit Namespace Object
   ```

2. Click on [Create], and select [Printer] from the menu that appears.
3. Specify a name for your printer. Note that you cannot include spaces in the printer name. For example, you can name a printer Washington-Post, but not Washington Post. Compare the display of your Namespace Editor to figure 1.
4. Move the mouse to the Site attribute, click on *Site*, and specify the name of your site.
5. Move the mouse to the Pretty Name attribute, click on *Token*, and specify the name of the printer.
6. Move the mouse to the Host attribute, click on *Host*, and specify the name of the host to which you cabled the printer. If you are spooling the printer from a MacIvory or a UX-family machine, specify the name of the embedded Symbolics host.
7. If you customize the communication options for your printer, specify the appropriate interface options and their values in the Interface Options field. For more information, see the section "Parameters for Serial I/O".

For more information on the printer attributes defined in printer namespace objects, see the section "Attributes for Objects of Class "Printer"".

Figure 2 displays the Washington-Post's partially filled in namespace object at a site named SCRC, cabled to a host named Riverside.

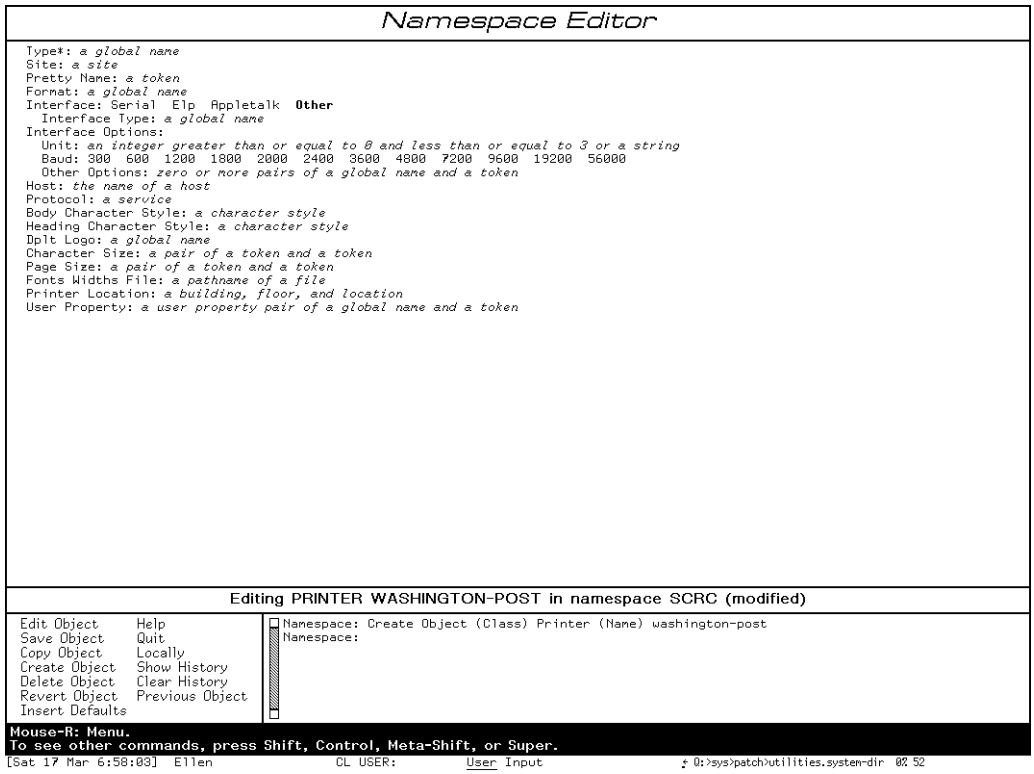


Figure 118. Namespace Editor editing PRINTER WASHINGTON-POST

Creating a Printer Object

Depending on the printer you are installing, proceed to one of the following sections for creating a printer object:

- Creating an LGP2 Printer Object
- Creating an LGP3 Printer Object
- Creating an ASCII Printer Object
- Creating a DMP1 Printer Object

Creating an LGP2 Printer Object

After you build a skeleton namespace object for your printer, you can create an LGP2 printer object following these steps:

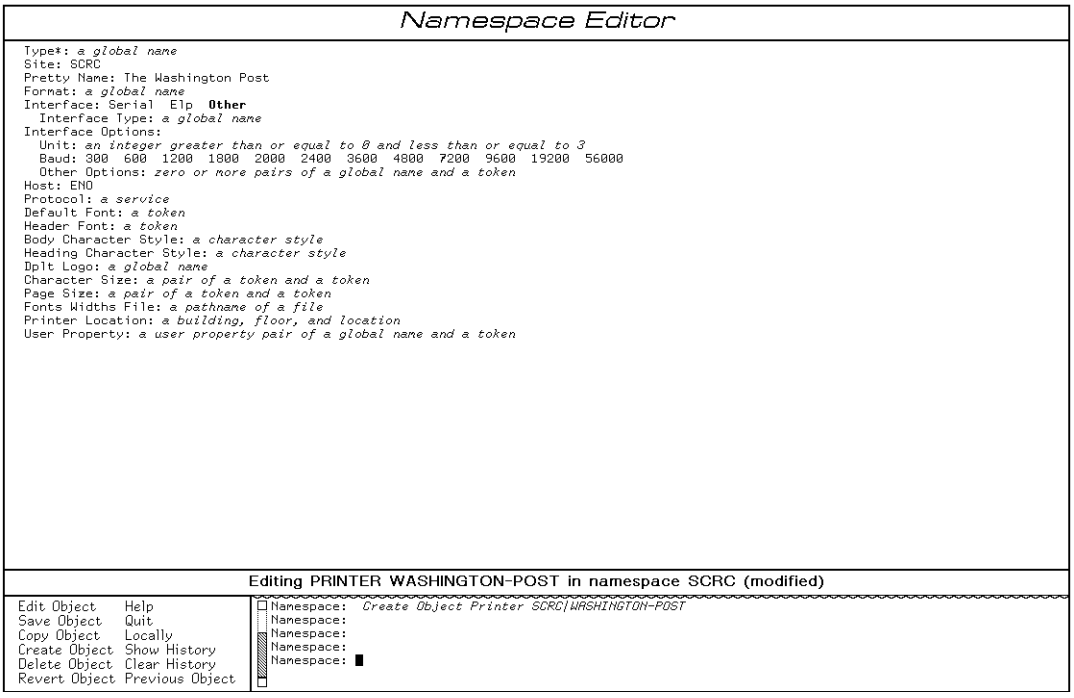


Figure 119. Washington-Post's Namespace Object

1. Move the mouse to the Type attribute, click on *Global-Name*, and specify `lgp2`.

2. Move the mouse to the Interface field and click on Serial.

3. Move the mouse to the Interface Options field. Locate the Baud field and specify the appropriate baud rate, for example 9600 or 1200. You can set the LGP2's baud rate by adjusting the small metal rotary switch (**mode switch**) on the back of the LGP2. See the section "Specifying the Switch Settings".

4. If you are connecting an LGP2 to an XL400, you must specify serial handshake protocol options. Move the mouse to the Interface Options field and click on *Zero or more pairs of a global name and a token*. Specify the following option pairs:

```
NUMBER-OF-DATA-BITS 8 PARITY :NONE XON-XOFF-PROTOCOL YES
```

5. Move the mouse to the Unit field and click on the unit number (you can determine the unit number by looking at the label on the serial port to which you cabled the printer). For UX-family hosts, the unit number is the appropriate serial device specification for the Sun system. For more information, see the section "Cabling the Printer".

6. You can change the character styles for your printer by specifying a character style in the Body Character Style field.

7. Click on the [Save] menu item and wait for a message confirming that the object is saved.

8. Click on the [Quit] menu item.

Creating an LGP3 Printer Object

After you build a skeleton namespace object for your printer, you can create an LGP3 printer object following these steps:

1. Move the mouse to the Type attribute, click on *Global-Name*, and specify `lgp3`.

2. Move the mouse to the Interface field and click on Serial.

3. Move the mouse to the Interface Options field. Locate the Baud field and specify the appropriate baud rate, for example 9600 or 1200. You can set the LGP3 baud rate by adjusting the small dual dip switch (**mode switch**) on the back of the LGP3. See the section "Specifying the Switch Settings".

4. Move the mouse to the Unit field and click on the unit number (you can determine the unit number by looking at the label on the serial port to which you cabled the printer). For UX-family hosts, the unit number is the appropriate serial device specification for the Sun system. For more information, see the section "Cabling the Printer".

5. You can change the default character styles for your printer by specifying a character style in the Body Character Style field.

6. Click on the [Save] menu item and wait for a message confirming that the object is saved.

7. Click on the [Quit] menu item.

None of the printer attributes other than Body Character Style are supported by the LGP3 printer support software.

Creating an ASCII Printer Object

In order to create an ASCII printer object, follow these steps:

1. Move the mouse to the Type attribute, click on *Global-Name*, and specify `ascii`.

2. Move the mouse to the Interface field and click on Serial.

3. Move the mouse to the Interface Options field. Locate the Baud field and specify the appropriate baud rate, for example 9600 or 1200. For more information, see the section "Specifying the Switch Settings".

4. Specify page size by using the page size attribute. The default page size is for a printer using wide carriage paper. If your printer uses smaller paper (8 1/2 inches by 11 inches), set the page size attribute to:

```
PAGE-SIZE 66 80
```

5. You must also specify additional Interface Options matching the serial communication parameters specified for the printer. For a list of parameters, keyword symbols, and default values for the serial I/O facility, see the section "Parameters for Serial I/O" in *Reference Guide to Streams, Files, and I/O*.

   The Interface Options fields are the same as the keywords for making serial streams, but without the preceding colon. You can specify the options in any order. Following is a list of several Interface Options. Experiment with different combinations of these options until the printer works correctly.

   Number-of-data-bits
   One of the following fixnums represents the number of data bits: **5, 6, 7** (default), or **8**. If your printer accepts 8 bits, you do not have to specify the **output-xoff-character** or the **output-xon-character**. Otherwise, check to see whether the system recognizes the **output-xoff-character** with a value of #o23. If not, change the value to #o223. If changing the **output-xoff-character** does not solve the problem, change the **output-xon-character** character from #o21 to #o221.

   Parity
   Specifying a value of **nil** transmits no parity. Specifying **:even** (default) transmits even parity. Specifying **:odd** transmits odd parity.

   Xon-xoff-protocol
   The default value is **nil**, but most printers require a value of **t**.

6. Move the mouse to the Unit field and click on the correct unit (serial port) number. For UX-family hosts, the unit number is the appropriate serial device specified for the Sun system. For more information, see the section "Cabling the Printer".

7. Click on the [Save] menu item and wait for a message confirming that the object is saved.

8. Click on the [Quit] menu item.

Note that the Page Size attribute is the only printer attribute supported by the ASCII printer support software.

Creating a DMP1 Printer Object

After you build a skeleton namespace object for your printer, you can complete the DMP1 printer object following these steps:

1. Move the mouse to the Type attribute, click on *Global-Name*, and specify `dmp1`.

2. Move the mouse to the Interface field and click on Serial.

3. Move the mouse to the Interface Options field. Locate the Baud field and specify the appropriate baud rate, for example 9600 or 1200. If you used the recommended switch settings for a DMP1 printer, specify 9600 baud. For more information, see the section "Specifying the Switch Settings".

4. If you connect the DMP1 to an XL400, you must specify serial handshake protocol options. Move the mouse to the Interface Options field and click on *Zero or more pairs of a global name and a token*. Specify the following option pairs:

```
NUMBER-OF-DATA-BITS 8 PARITY :NONE XON-XOFF-PROTOCOL YES
```

5. Move the mouse to the Unit field and click on the unit number (you can determine the unit number by looking at the label on the serial port to which you cabled the printer). For UX-family hosts, the unit number is the appropriate serial device specification for the Sun system. For more information, see the section "Cabling the Printer".

6. You can change the character styles for your printer by specifying a character style in the Body Character Style field.

7. Click on the [Save] menu item and wait for a message confirming that the object is saved.

8. Click on the [Quit] menu item.

Note that the DMP1 printer support software only supports the Body Character Style attribute.

Associating the Printer With its Controlling Host

Registering a printer includes assigning a controlling host for the printer (often called the *spooler* for that printer). The host acting as the spooler makes the printer accessible to multiple users simultaneously.

The controlling host is usually the same host that you specify when you create the printer object. When you register a printer connected to a UX-family or a MacIvory serial port, the Symbolics host is the controlling host. You can specify a controlling host other than the machine to which you physically connect the printer, but Symbolics recommends that you physically connect the printer to the controlling host.

You can enable spooling services for a printer by installing the Print Spooler on the controlling host. If you do not install the Print Spooler, you can send only one job at a time to the printer. For more information, see the section "Defining the Print Spooler".

Use the Namespace Editor for editing an object representing the host spooler for the new printer.

1. Enter the Namespace Editor by specifying the following command from a Lisp Listener:

   ```
   Edit Namespace Object
   ```

2. Click on [Edit Object]. A menu appears asking you to:

   ```
   Enter a namespace object
   ```

   Specify `host`, and name the spooling host. Note that you have to press `RETURN` to enter the host and name, and then press `END` to enter the namespace object.

3. After the host object is read in, move the mouse to the Spooled-Printer attribute. Specify the name of the printer and press `RETURN`. Note that you only add the Spooled-Printer attribute to the host object of the print server, not to any other machines at the site. The Home Directory field appears.

   The new printer stores queued requests on disk prior to printing them. Such requests are stored in a subdirectory of the directory associated with the spooler. The directory associated with the printer is named >Print-Spooler>, and the subdirectory is named after the printer. For example, print requests are stored in >Print-Spooler>Washington-Post>. Both the top-level directory >Print-Spooler> and the subdirectory >Print-Spooler>Washington-Post> are created when you start the Print Spooler. If the host spools multiple printers, try to keep all printer home directories in one directory, such as >Print-Spooler>.

   Specify the name of the directory in the spooled-printer attribute of the host object:

   ```
   host:>print-spooler>washington-post>
   ```

For more information, see the section "The Spooled Printer Host Attribute".

4. You have to add appropriate service attributes to the controlling host. These service attributes specify that the spooling host supports the services necessary for hardcopying to the connected printers. Move your mouse to a service field appearing as follows:

   Service Field: *a triple of service, medium, and protocol*

   You can specify the beginning letters of a service, medium, or protocol and press SPACE to complete the correct name. Also note that pressing HELP in this field gives you information about your choices.

   If you are associating a printer with a UNIX host, see the section "Configuring a Namespace to Use UNIX Print Spoolers".

5. Click on *a triple of service, medium, and protocol*. Specify HARDCOPY as the service name, CHAOS as the network by which requests are queued, and Printer-Queue as the contact name. For example:

   ```
   HARDCOPY CHAOS PRINTER-QUEUE
   ```

   Press RETURN after typing the last service.

6. In the new service field created below the one you just edited, click on *a triple of service, medium, and protocol*. Specify PRINTER-QUEUE-CONTROL as the service name, CHAOS as the network by which requests are queued, and PRINTER-QUEUE as the contact name. For example:

   ```
   PRINTER-QUEUE-CONTROL CHAOS PRINTER-QUEUE
   ```

   Press RETURN after typing the last service.

7. In the new service field created below the one you just edited, click on *a triple of service, medium, and protocol*. Specify PRINTER-CONTROL as the service name, CHAOS as the network by which requests are queued, and PRINTER-QUEUE as the contact name. For example:

   ```
   PRINTER-CONTROL CHAOS PRINTER-QUEUE
   ```

   Press RETURN after typing the last service. The resulting triplets appear as follows in the namespace editor:

   ```
   Service: HARDCOPY CHAOS PRINTER-QUEUE
   Service: PRINTER-QUEUE-CONTROL CHAOS PRINTER-QUEUE
   Service: PRINTER-CONTROL CHAOS PRINTER-QUEUE
   ```

   If you are using TCP on all computers at your site, you can specify TCP as the medium instead of CHAOS. If you are using both TCP and CHAOS, add each service triplet twice: once with CHAOS as the medium and once with TCP as the medium.

8. If some machines at your site are running Genera 7.2, and others are running Release 6.1, add these additional services:

- Find the last "Service" field.

  Click on *a triple of service, medium, and protocol*. Specify HARDCOPY as the service name, CHAOS as the name of the network by which requests are queued, and lgp as the contact name. For example:

  ```
  HARDCOPY CHAOS LGP
  ```

  Press RETURN after typing the last service.

- Click on *a triple of service, medium, and protocol* in the new service field created below the one you just edited. Specify hardcopy-status as the service name, chaos as the network by which requests are queued, and lgp-queue as the contact name. For example:

  ```
  HARDCOPY-STATUS CHAOS LGP-QUEUE
  ```

  Press RETURN after typing the last service.

9. Click on the [Save] menu item, and wait for a message confirming that the object is saved.

10. Click on the [Quit] menu item.

Specifying Whether a Banner Page Prints

You can specify whether banner pages print by adding the following option to the Spooled Printer attribute of the host:

```
PRINT-COVER-PAGE
```

Specifying PRINT-COVER-PAGE YES enables the printing of banner pages. Specifying PRINT-COVER-PAGE NO prevents the printing of banner pages.

Note that you can control the printing of banner pages for individual jobs using the Command Processor command Hardcopy File.

For more information concerning the print spooler options host attribute, see the section "The Print Spooler Options Host Attribute".

Editing the Site Object

You can specify a default printer following these steps:

1. Enter the Namespace Editor by specifying the following command from a Lisp Listener:

   ```
   Edit Namespace Object
   ```

2. Click on [Edit Object]. A menu appears asking you to specify a name-space object. Specify site, and then specify the name of the site you are

editing. Note that you have to press RETURN in order to enter the site name.

3. Move the mouse to the attribute Default Printer or Default Bitmap Printer, depending on the type of printer you have at your site. Specify the name of the printer and press RETURN.

For more information, see the section "The Default Printer Site Attribute". Additionally, see the section "The Default Bitmap Printer Site Attribute".

You can override the default printer for a site by:

- Resetting the printer for a host using the Printer and/or Bitmap Printer attributes. For more information: See the section "The Bitmap Printer Host Attribute". See the section "The Printer Host Attribute".

- Using the Set Printer command or by setting appropriate variables in you init file. For more information, see the section "Customizing Hardcopy Facilities".

Defining the Print Spooler

When defining the print spooler, read the following:

- Print Spooler Overview

- Adding an Internet Domain Name for the Local Namespace

- Loading the Print Spooler Software

- Specifying Print Spooler Options

- Controlling the Print Spooler

Print Spooler Overview

The Print Spooler organizes print requests from multiple users using the same printer. The Print Spooler sends print requests to the printer in the order in which they are received. Due to the fact that a printer prints only one request at a time, the Print Spooler stores additional print requests in the **home-directory** associated with the printer. Note that one print spooler can handle several printers.

The Print system defines the Print Spooler. You must load and save the Print Spooler into the world running on the host spooling the printer (print server.) After you load and save the Print Spooler in a world, it automatically starts when you enable the spooling host services (usually after booting).

The Print Spooler maintains a printer *queue* for each printer. You can verify the state of the queue using the "Show Printer Status Command". For printer queue information, see the section "Managing the Print Queue".

Adding An Internet Domain Name for the Local Namespace

The Print Spooler requires an Internet Domain name for the local namespace object. The Internet domain name is part of the mail-name of your local host, to which the Print Spooler sends notifications about print requests.

Not specifying an Internet domain name causes a print spooler error because the spooler sends a message to an unknown host (appearing as *host.site*.Dialnet.Symbolics.COM). The error occurs when the print spooler attempts to send a notification to the default mail-name of the host from which a print request is submitted; the mail-name cannot be parsed by **net:parse-host**. An example of the error message follows:

```
Error: "Bald-Eagle.Dialnet.Symbolics.Com" is not the name of a known host.
       No host with that name is already known by the local machine,
       but it is not certain that there is no such host.
       Some namespace server did not respond.
```

If you already have an Internet domain name for your namespace or for each of your individual hosts, you have met the Print Spooler requirement. If you do not yet have an Internet domain name for your namespace, you must add one before using the Print Spooler. For more information about Internet domain names, see the section "Internet Domain Names". For information about namespace objects, see the section "Concepts of the Namespace System".

Follow these steps for adding an Internet domain name to your local namespace object:

1. Enter the Namespace Editor for the namespace object by specifying the following CP command:

   ```
   Edit Namespace Object Namespace namespace-name
   ```

   where *namespace-name* is the name of the namespace in which your hosts are registered (usually the same name as your site).

2. Click Left on *token* in the field labeled "Internet Domain Name:"

3. If your namespace does not have a specific domain address for the ARPA Internet, Symbolics adopts your organization as part of the Symbolics Dial Network. In this case, the domain name for your organization is *namespace-name*.Dialnet.Symbolics.COM. For example, the Cobolics corporation owns several machines residing at a site named Cobolics. The Internet domain name is then Cobolics.DialNet.Symbolics.COM. Make sure the *namespace-name* is the same as your site name. Specify your namespace name and press RETURN.

If your namespace already has an assigned domain name in the ARPA Internet, specify that name and press RETURN. For example, Yoyodyne Propulsion Systems is a government aerospace contractor connected to the ARPA Internet via a gateway host and IMP. The administrators of the ARPA Internet assigned Yoyodine Propulsion Systems the domain name Yoyodyne.COM., which you specify as the Internet Domain name in the namespace.

4. Click on [Save] in the menu.

5. You can now leave the Namespace Editor by selecting any other activity with the SELECT key or by clicking on [Quit].

Loading the Print Spooler Software

You must load the Print Spooler software on hosts acting as print spoolers. You can load the Print Spooler software by specifying the following command from a lisp listener:

```
Load System "PRINT"
```

If you cannot load the print system, make sure you have restored the print system from the Genera 8.0 tapes. The Print system files for the Print Spooler are stored in the SYS:PRINT; directory.

Saving a World After Loading the Print Spooler

You can save a world after loading the Print Spooler to avoid loading the Print Spooler each time you reboot. When you boot a world containing the Print Spooler, the Print Spooler automatically starts (unless you specify otherwise by setting the Server Machine attribute of the spooler host to YES). For more information, see the section "The Server Machine Host Attribute".

For the procedure on building and saving a world with the print spooler loaded, see the section "Making Customized Genera User and Server Worlds".

For more infomation on "server" hosts, see the section "What is a File Server?".

Manually Starting the Print Spooler

If you do not save a world with the print spooler loaded, you must manually start the Print Spooler. Follow these steps for starting the Print Spooler:

1. Press SELECT S to select the Print Spooler log window

2. Specify the command Start Print Spooler at the print spooler command prompt:

```
Print Spooler command: Start Print Spooler
```

Specifying Print Spooler Options

Specifying Whether a Banner Page Prints

You can specify whether banner pages print by adding the following option to the `Spooled Printer` attribute of the host:

```
PRINT-COVER-PAGE
```

Specifying `PRINT-COVER-PAGE YES` enables the printing of banner pages. Specifying `PRINT-COVER-PAGE NO` prevents the printing of banner pages.

Note that you can control the printing of banner pages for individual jobs using the Command Processor command Hardcopy File.

For more information concerning the print spooler options host attribute, see the section "The Print Spooler Options Host Attribute".

Specifying Printer Timeout Warnings

You can use the print spooler option Timeout Notifications for determining whether a printer is operating. You can set the timeout notifications by adding the following option to the *Spooled Printer* attribute of the host:

```
WARNINGS-AFTER-TIMEOUT
```

Specifying `WARNINGS-AFTER-TIMEOUT YES` enables the timeout notifications (default.) Specifying `WARNINGS-AFTER-TIMEOUT NO` prevents the spooler from sending the timeout notifications.

The spooler sends notifications when printers are not responding. For example:

```
2/6 08:35:26 From ALLEGHENY: Printer "Wall Street Journal" needs
             intervention: The printer has not responded in 16 minutes while
             printing the current request.
             You may want to see if the printer is still operating.
```

By default, the first two notifications occur 4 and 8 minutes after the printer stops operating. Subsequent notifications occur at sixteen and thirty-two minutes. You can control when the first two timeout notifications occur by adding the following option to the *Spooled Printer* attribute of the host:

```
PRINTER-OUTPUT-TIMEOUT
```

When specifying a time interval, note that an integer represent a number of seconds. You can specify minutes, for example, "10 minutes". For more information concerning valid time intervals, see the section "Dates and Times".

Reasons for Timeouts

Timeout notification generally indicate a recoverable error, such as running out of paper. If you are using a PostScript printer (for example, an LGP2 or LGP3), timeout notifications can occur when you are executing a long PostScript program. If you accidentally disconnect the printer or the printer enters an unknown state, reset the printer using the "Reset Printer Command".

For more information concerning the print spooler options host attribute: See the section "The Print Spooler Options Host Attribute".

Controlling The Print Spooler

You can manipulate the Print Spooler through the Print Spooler Log window. In order to access this window, press SELECT S. The top part of the window displays log entries for print jobs currently printing and logs the entries into the current log file (located in the spooler's directory ">Print-Spooler>" on the local host). Each time you start the printer, the spooler creates a new log file. The bottom part of the window displays the print spooler command prompt. The Print Spooler command prompt enables you to specify commands pertaining to the print spooler, as well as all commands normally found in the Lisp Listener. For more information on the printer and printer queue commands, see the section "Managing the Print Queue".

If you have just loaded the Print Spooler software or have previously halted the Print Spooler, you can start the Print Spooler by clicking on [Start Print Spooler] in the command menu. You can stop the Print Spooler by clicking on [Stop Print Spooler] in the command menu. Note that a request printing at the time the Print Spooler stops automatically reprints the next time the Print Spooler starts. Note that after you halt the Print Spooler, you can continue sending requests to spooled printers, but the printers print the requests only after you restart the spooler.

When logging out or rebooting the print server host, you must halt the printer to assure that the printer log closes properly. You can use either the Halt Printer Command or the "Disable Services Command".

Installing and Using UNIX Printers From Genera

Genera 8.1 supports printers spooled from UNIX systems. The Symbolics printer software uses the UNIX lpd protocol for submitting formatted hardcopy requests to PostScript or DMP1 printers. To use this feature, you must first configure the printer namespace object on the Genera side and edit the /etc/hosts.lpd file on the UNIX side.

More information is available about editing the printer's namespace object. See the section "Attributes for Objects of Class "Printer"". See the section "Using the Namespace Editor".

Configuring a Namespace to Use UNIX Print Spoolers

The following example shows the namespace object of the printer PRAVDA connected to a UNIX host FROSTED-FLAKES:

```
PRINTER PRAVDA
TYPE LGP2
SITE AI
HOST FROSTED-FLAKES
PRETTY-NAME Pravda
```

If you have some special filter for output sent directly to the printer, set the user property **lpd-file-code-char** to the correct value. See the user property **lpd-file-code-char**.

Here's an example of a namespace object of the UNIX host FROSTED-FLAKES to which the printer PRAVDA connects. The namespace attributes important to UNIX-LPD are in boldface:

```
HOST FROSTED-FLAKES
SYSTEM-TYPE UNIX42
SITE AI
MACHINE-TYPE SUN-3
ADDRESS INTERNET 128.52.32.19
SERVICE EXPAND-MAIL-RECIPIENT TCP SMTP
SERVICE FILE TCP TCP-FTP
SERVICE FILE UDP TFTP
SERVICE FILE UDP NFS
SERVICE LISPM-FINGER UDP LISPM-FINGER
SERVICE LOGIN TCP TELNET
SERVICE MAIL-TO-USER TCP SMTP
SERVICE SEND TCP SMTP
SERVICE SHOW-USERS TCP ASCII-NAME
SERVICE HARDCOPY TCP UNIX-LPD
SERVICE PRINTER-CONTROL TCP UNIX-LPD
SERVICE PRINTER-QUEUE-CONTROL TCP UNIX-LPD
SPOOLED-PRINTER PRAVDA
```

The service namespace attribute tells the Genera generic networking system to use the UNIX `lpd` hardcopy protocol for the `HARDCOPY`, `PRINTER-CONTROL`, and `PRINTER-QUEUE-CONTROL` services. The spooled-printer attribute tells the hardcopy system that FROSTED-FLAKES is a Print Spooler host for the printer PRAVDA.

The Symbolics printer software formats the output based on the type of printer you specify in the printer namespace object. For example, if you are spooling a PostScript printer from a UNIX system, you must specify a printer object type LGP2 or LGP3. After you specify the printer object type, the Genera hardcopy facilities can produce and submit PostScript to the UNIX spooler.

Note: If you are using a 3600-family machine, you must have the systems RPC, Embedding-Support, and UX-Support loaded before you can use a printer spooled from a UNIX machine.

UNIX Configuration for Using UNIX Printers From Genera

The `lpd` daemon on a UNIX host does not accept any hardcopy requests from hosts not listed in the `/etc/hosts.lpd` file. Add the name of your Symbolics machine to this file to enable the use of the basic set of hardcopy commands (Hardcopy File, Show Printer Status, and the Delete Printer Request).

lpd-file-code-char *char* *User Property*

A user property for printer namespace objects. The value of *char* tells UNIX which filter selecting command to use when printing data from Genera. The default filter is specified with `-f`, the default value for this user property.

To send a special filter for output directly to the printer, set the value of *char* to match the letter that corresponds to that filter. For example, if you have an Imagen printer connected to a UNIX machine, you must give the `-v` option to `lpr` to send imPress files to it. In this case, the value for the user property **lpd-file-code-char** will be `v`, corresponding to the `-v` option.

Installing and Using Printers with MacIvory

Using an Appletalk Printer From the Macintosh and Genera

- Printing via AppleTalk is supported only for Postscript printers.
- Printing via AppleTalk can not be used with our Print Spooler.

To use an AppleTalk printer do the following:

1. Select your target printer on the Macintosh side through the Chooser.
2. Create a printer object in the namespace that identifies your MacIvory as the host to which the printer is connected.
3. Set the default printer for your MacIvory to the printer object you just created. You can do this either in the namespace to make the effect permanent or with the Set Printer CP command.

You can now use the printer from Lisp using the normal hardcopy facilities of Genera.

The use of separate printer objects is required even if several MacIvories are using the same AppleTalk printer. Of course, each printer object in the namespace must have a unique name and pretty name. Therefore it is a good idea to include the name of the MacIvory in the printer's names.

For example,

Showing PRINTER SOUR-CREAM-LOS-ANGELES-TIMES in namespace SCRC:

```
Type: LGP2
Site: SCRC
Pretty Name: The Los Angeles Times for Sour Cream
Interface: AppleTalk
Host: SOUR-CREAM
Printer Location: SCRC 2 Outside KMP's office
```

The `Type` attribute should be LGP2, LGP3, or POSTSCRIPT depending on the type of printer. LGP2 corresponds to the old LaserWriter and LaserWriter Plus. LGP3 corresponds to the newer LaserWriter IINT and LaserWriter IINTX. LGP3 also works for Apple's latest printer, the Personal LaserWriter NT. Use POSTSCRIPT for non-Apple Postscript printers (for example, HP).

In the namespace object for the MacIvory, make sure that the `BITMAP-PRINTER` attribute is empty. Otherwise, Lisp tries to invoke the Print Spooler, which does not work.

If your Macintosh is running MultiFinder, it's a good idea to enable background printing in the Chooser when you select your AppleTalk printer. Otherwise, whenever you print something in Lisp, Genera stops while printing takes place. (You see the familiar Macintosh modal dialog boxes describing the print process appear in front of the Genera screen.)

Using the Print Spooler with Genera

To spool an LGP2/LGP3 printer from a MacIvory machine, you need one 8-pin-male-to-25-pin-male cable with a null modem in it. Alternatively, you can use any combination of cables that provides you with this configuration.

1. Plug the printer into one of the serial ports on the back of the Macintosh computer. You can run the printer from the modem port (`UNIT 0`) or the printer port (`Unit 1`).

2. Edit the printer's Namespace Object as follows:

   ```
   Interface: SERIAL
   Interface Options: UNIT n BAUD 9600 PARITY :none NUMBER-OF-DATA-BITS 8
           XON-XOFF-PROTOCOL yes
   User Property: DTR-VALID nil
   ```

   More information is available about editing the printer's namespace object. See the section "Attributes for Objects of Class "Printer"". See the section "Using the Namespace Editor".

3. Set the baud rate to 9600 on the LGP2/LGP3 printer.

Note: If you need more information about Macintosh serial ports, check the Unit specification conventions in the MacIvory User's Guide.

For additional information about setting up the print spooler and the LGP2/LGP3 printer,see the section "Installing a Printer".

See the section "Uncrating the Printer".

See the section "Specifying the Switch Settings".

See the section "Loading the Hardcopy and Printer Support Tape".

See the section "Registering a Printer".

Note: If you boot the MacIvory while the print spooler is running, you must then restart the Macintosh computer. Otherwise, the serial stream is left open (and attempts to restart the print spooler will fail with an "unable to access the serial stream" error).

Using PostScript Printers

The LGP2 and LGP3 printer support has been remodularized in Genera 8.0 to separate out some of the LaserWriter-specific routines so that it can be used more generally with arbitrary PostScript printers. As much as possible, this has been done without changing the names of various symbols that contained the string LGP2 and LGP3. This should make the change nearly invisible, even to application extensions.

The format of bytes sent to a printer is determined by the Type attribute of the printer namespace object. The type `POSTSCRIPT` is now supported for basic PostScript printers. Additionally, the types `LGP3`, `LASERWRITER`, and `LASERWRITER-2` are accepted as synonyms for `LGP2`. The differences between `POSTSCRIPT` and `LGP2` are covered in detail below.

The hardcopy file output type `PS` is now accepted. A .ps file written by the hardcopy system will be mostly the same as the .lgp2 file previously written, except that the pages will not be reversed and no allowance will be made for font memory. See below for more details.

Since there are still differences among various PostScript printers, and Symbolics does not have access to most of them, there is an extension mechanism for PostScript printer types.

lgp::define-postscript-printer-type *type* &key *(:display-device-type* **lgp::*lgp2-postscript-printer***) *(:default-body-character-style* **'(:fix :roman :normal)**) *(:default-heading-character-style* **'(:heading :bold :normal)**) *(:edge-label-font-name* **"Helvetica"**) *(:input-formats* **(:postscript)**) *:output-format (:xon-xoff-protocol* **t**) *(:dtr-valid* **t**) *:default-serial-interface-options (:default-print-backwards* **t**) *:get-printer-name-command :set-printer-name-command :load-adobe-patch (:load-adobe-error-handler* **t**) *:font-memory-threshold (:pixels-per-inch* **300**) *(:after-document-timeout* **lgp::*default-after-document-timeout***) *:print-cover-page-before :color-p* *Function*

Defines a printer type based on PostScript. The allowed keywords include:

:display-device-type The character style device type to use. If the printer uses entirely different font names, you must make a new set of font mappings and specify them here. The default is **lgp::*lgp2-postscript-printer***.

:default-body-character-style and *:default-heading-character-style*
The character styles used by default for Hardcopy File and related commands. It is usually better to change the mappings than to change these. The default for *:default-body-character-style* is **(:fix :roman :normal)**. The default for *:default-heading-character-style* is **(:heading :bold :normal)**.

:input-formats A list of hardcopy file formats (as defined by **hci:define-hardcopy-format**) that this printer will accept. If you have a different format for your printer, specify it here. The default is **(:postscript)**.

:output-format A list of hardcopy file formats (as defined by **hci:define-hardcopy-format**) that the other options correspond to. If you have a different format for your printer, specify it here in addition to the **hci:define-hardcopy-format**. The default is **nil**.

:xon-xoff-protocol The printer uses XON/XOFF for flow control when connected via a serial line. The default is **t**.

:dtr-valid The printer asserts DTR over its RS-232 cable when connected via a serial line. The default is **t**.

:default-print-backwards
Unless otherwise specified, a stream to this printer should reverse the pages before sending so that they will collate in the proper order. If the printer itself, or a spooler on a remote host, reverses the pages, you disable this. The default is **t**.

:get-printer-name-command and *:set-printer-name-command*
PostScript code to be used to read and write the printer name stored internally in the device. If **nil**, no attempt will be made to change the name when the print spooler starts up. The default for each is **nil**.

:load-adobe-patch If **t**, attempt to download the latest Adobe patch into the printer when starting the print spooler. This patch is probably LaserWriter-specific. The default is **nil**.

:load-adobe-error-handler
If **t**, attempt to download the improved Adobe error handler into the printer when starting the Print Spooler. This handler prints a page of error text whenever the printer gets an error interpreting a PostScript file. The error handler itself is generic PostScript code, and should work in printers other than the LaserWriter, especially if they run Adobe software. However, sometimes when a LaserWriter is connected to both a Macin-

tosh and a serial line, there is enough line noise to cause occasional errors which are better ignored than reported. In this case, the new error handler can be disabled. The default is **t**.

:font-memory-threshold

If non-**nil**, the amount of memory available for downloading fonts. When printing files with many fonts in the whole document, it is often not possible to fit them all at the head of the file. The hardcopy system can split them into bunches per page and load them incrementally (making the PostScript file larger, but using less memory). This is most often necessary with TeX output converted from .dvi files. The number for any given printer can really only be determined empirically. You can set the flags **lgp::*show-vmstatus-on-each-page*** and **lgp::*show-font-memory-usage*** for debugging to see how much is ordinally available at the start of a page. The default is **nil**.

:pixels-per-inch The resolution of the device. Unfortunately, this does not yet control the selection of downloadable bitmap fonts, such as the Bitstream fonts used by Symbolics Concordia. The default is 300.

:after-document-timeout

The amount of time to wait after sending a document for the printer to resume responding. If the printer is very slow, or documents are very complex, this may need to be increased. The default is **(* 60 60 10)**, which is 10 minutes.

:color-p The device natively supports printing in color. Printing from the images system uses this to decide whether or not to first reduce the image to gray scale. The default is **nil**.

The definition for an unspecific `POSTSCRIPT` type is

```
(define-postscript-printer-type :postscript
  :default-print-backwards nil
  :output-format :postscript)
```

which allows for the .ps file type and disables reversing pages.

The definition for the `LaserWriter` type is:

```
(define-postscript-printer-type :lgp2
  :input-formats (:postscript :lgp2)
  :output-format :lgp2
  :get-printer-name-command *lgp2-printername-command*
  :set-printer-name-command *lgp2-setprintername-command*
  :load-adobe-patch t
  :font-memory-threshold *lgp2-font-memory-threshold*
  )
```

which allows for the .lgp2 file type, setting the printer name in the statusdict, loading the Adobe patch, and an empirically determined font memory threshold.

It is also possible to set these options on a per-printer basis using the User Property List attribute of the printer namespace object. The indicator (the global name) is the keyword shown above, and the value (the token) is the printed representation of the desired value. Since this mechanism requires parsing and does not have very good error checking, it is not recommended for wholesale redefinition. It is better to use **lgp::define-postscript-printer-type**. However, it is useful for small modifications, such as changing whether page collation is done by the spooler or the hardcopy system.

Since there is still a fair degree of possible variance among PostScript spoolers and printers, it may be that more options are required. If you have trouble with a previously unsupported PostScript printer, you should contact customer support. Otherwise, there is no way it can become supported, since Symbolics almost certainly does not have one.

Managing the Print Queue

You can use the following commands for managing the print queue:

- Show Printer Status Command
- Delete Printer Request Command
- Restart Printer Request Command
- Halt Printer Command
- Start Printer Command
- Reset Printer Command

Show Printer Status **Command**

Show Printer Status *printer*

Displays the print queue for the specified printer or printers.

printer — The name of a printer or printers (separated by commas) whose print queue you are displaying. Specifying All displays the queues for all printers at your site. The default is your current printer. If your text printer and your bitmap printer are different, the command uses your text printer as the default for Show Printer Status.

The display of requests is mouse sensitive, enabling you to click on select arguments when using either the Delete Printer Request or Restart Printer Request commands.

Note that this command is also available in Zmail as m-X Show Printer Status.

Delete Printer Request **Command**

Delete Printer Request *printer-request*

Deletes the specified print request from the print queue.

printer-request A string specifying the printer and the request you are deleting. You can select the print request with the mouse from the display of the Show Printer Status command. For more information, see the section "Show Printer Status Command". You can also specify the printer name to which you are sending requests and use "c-? and c-/"for displaying all requests. For more information, see the section "Completion in the Command Processor".

keywords :Confirm

:Confirm {Yes, No} Whether to request confirmation when you delete a request currently printing. The default is Yes.

Restart Printer Request **Command**

Restart Printer Request *printer printer-request keywords*

Restarts a print request that you stopped before completion. For requests currently printing, the printer resets and the request prints from the beginning. For requests on hold, the printer requeues the request.

printer-request A string specifying the request. You can select the print request with the mouse from the display of the Show Printer Status command. You can also specify the printer name to which you sent the request and use "c-? and c-/" for displaying all requests. For more information, see the section "Completion in the Command Processor".

keywords :Extent, :Starting From

:Extent {Entire, Copy} The extent to which you are restarting a request. Entire refers to the whole request. Copy refers to a single copy.

:Starting From {*number*} The number of the copy after which the printer restarts. Note that you cannot use this keyword if you used the :extent keyword specifying Entire. The default is 0. *This option is currently not used.*

Halt Printer **Command**

Halt Printer *printer keywords*

Halts the specified printer. The printer does not print any requests until you start it using the "Start Printer Command".

printer	The name of the printer you are halting.
keywords	:Confirm, :Disposition, :Reason, :Urgency
:Confirm	{Yes, No} Whether to ask for confirmation. The default is Yes.
:Disposition	{Delete, Hold, Restart} Enables you to specify whether the system delete, hold, or restart the print request. The Delete option deletes the request from the queue. Hold retains the request in the queue but does not print it when the printer restarts. Restart automatically prints the interrupted request from the beginning when the printer restarts. The default is Hold.
:Reason	{*string*} Enables you to specify the reason for the shutdown. This option appears in the display of the Show Printer Status command and in the Print Spooler log. The default message is "Printer *printer-name* being reset by *user-id*."

You can use the following keyword to control precisely when the printer halts:

:Urgency	{Asap, After-Current-Request, After-Next-Copy} Enables you to specify when the printer halts. Asap indicates that the printer halt and reset immediately. After-Current-Request indicates that printer halt after the current request finishes printing. After-Next-Copy indicates that the printer halt after the next copy of the request prints. After-Next-Copy is the same as After-Current-Reqest when the request is only for one copy. The default is Asap.

When you halt the printer, the Print Manager process enters a suspended state until you start it again.

Note that you can halt, start, or reset a spooled printer from any machine on the network.

Start Printer **Command**

Start Printer *printer*

Starts the specified printer printing its print queue after you halt it.

printer	The name of the printer you are starting.

Note that you can halt, start, or reset a spooled printer from any machine on the network. For more information, see the section "Halt Printer Command".

Reset Printer **Command**

Reset Printer *printer printer-request keywords*

Resets a spooled printer. You can use this command if your printer stops operating or if you have to stop the printer in the middle of a job. This command reestablishes communications with the printer and, when possible, sends a software "reset" command to the printer.

printer	The name of the printer you are resetting.
printer-request	(Optional) Resetting a printer in the middle of a print request results in the system displaying the request and asking you to confirm the reset command. You have to specify the name of the print request or select the print request with the mouse from the display of the Show Printer Status command in order to reset the printer immediately. For more information, see the section "Show Printer Status Command".
keywords	:Confirm, :Disposition, :Reason
:Confirm	{Yes, No} Whether to request confirmation that the printer is printing a request. The default is Yes.
:Disposition	{Delete, Hold, Restart} Enables you to specify whether the system delete, hold, or restart the print request. The Delete option deletes the request from the queue. Hold retains the request in the queue but does not print it when the printer restarts. Restart automatically prints the interrupted request from the beginning when the printer restarts. The default is hold.
:Reason	{*string*} The reason you are resetting the printer, which is added to the printer log. The default message is "Printer *printer-name* being reset by *user-id*."

Note that you can halt, start, or reset a spooled printer from any machine on the network.

Producing Hardcopy

You can produce hardcopy from:

- The Command Processor
- The System menu
- Zmacs

- Zmail
- Dired in the Editor
- The screen
- The File System Editor

For more information on using character styles in hardcopy, see the section "Using Character Styles in Hardcopy".

Note that you can also hardcopy topics from Document Examiner: See the section "Document Examiner".

Hardcopying from the Command Processor

Hardcopy File **Command**

Hardcopy File *pathname printer keywords*

Sends a file to a hardcopy device.

pathname — The pathname of the file you are printing. You can specify more than one file by separating the pathnames with commas.

printer — The printer on which you are printing the file. The default is the default hardcopy text printer (the value of **hardcopy:*default-text-printer***).

keywords — :Body Character Style, :Copies, :Delete, :Ending Page, :File Types, :Heading Character Style, :Notify, :Orientation, :Print Cover Page, :Running Head, :Starting Page, :Title

:Body Character Style
The character style used for printing the text of the file. You can also use this character style for merging character styles in the file. See the section "Understanding Character Styles".

:Copies — {*number*} The number of copies you are printing. The default is 1.

:Delete — {Yes, No} Enables you to specify whether the file deletes after printing. The default is No, not to delete.

:Ending Page — {*number*} The last page you are printing. This command defaults to the last page of the file. Note that the page character or form feed character identifies page breaks. The system treats text files without page breaks as a single page, although the file can use many sheets of paper.

Press format files contain form feeds or PAGE characters. Remember that these are physical pages and do not necessarily correspond to the page numbering appearing in the heading.

For example, the first page of a press file is a title page, the second page is numbered *i*, and the third page is numbered 1.

:File Types {ASCII, DMP1, LaserWriter, PostScript, Press, Text, XGP, use-canonical-type} The internal format of the contents of the file, to interpret for printing. The default is use-canonical-type, meaning that the type is determined from the extension to the file name.

:Heading Character Style
The character style used for the running head.

:Notify {Yes, No} Specifies whether to send a notification upon job completion. The default is Yes.

:Orientation {Landscape, Portrait} Specifies the paper orientation for the output. Portrait is left to right across the short dimension of the paper. Landscape is left to right across the long dimension of the paper. The default is Portrait.

:Print Cover Page {Yes, No} Specifies whether to print a cover page. The default is Yes.

:Running Head {None, Numbered} Specifies the type of running head that prints on the top of each page. The default is Numbered.

:Starting Page {*number*} The first page you are printing. This command defaults to the first page of the file. The page character or form feed character identifies page breaks. The system treats text files without page breaks as a single page, although the file can use many sheets of paper.

Press format files contain form feeds or PAGE characters. Remember that these are physical pages and do not necessarily correspond to the page numbering appearing in the heading. For example, the first page of a press file is a title page, the second page is numbered *i*, and the third page is numbered 1.

:Title {*string*} Specifies the title appearing on the cover page identifying the output. The default is "File: *pathname-you-specified* ".

Hardcopying From the System Menu

You can produce hardcopy by clicking on [Hardcopy] in the System menu. The Accept Variable Values menu appears, enabling you to specify the pathname of the file you are hardcopying, and to select the printer, character style, and the other parameters offered by the Hardcopy command (see Figure 84).

Hardcopying From Zmacs

You can use the following commands when hardcopying from Zmacs:

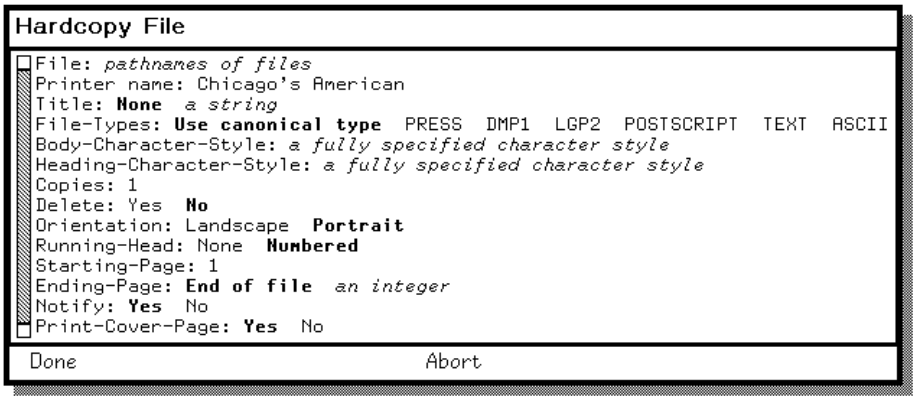


Figure 120. The Hardcopy Menu

- Hardcopy Region Command
- Hardcopy Buffer Command
- Hardcopy File Command
- Kill or Save Buffers Command

Hardcopy Region Command

Hardcopy Region (m-X)

Sends a region's contents to the local hardcopy device for printing.

For full information on Genera hardcopying, see the section "How to Get Output to a Printer".

Hardcopy Buffer Command

Hardcopy Buffer (m-X)

Prompts for the name of a buffer and then prints the specified buffer on the local hardcopy device.

For full information on Genera hardcopying, see the section "How to Get Output to a Printer".

Hardcopy File

Hardcopy File (m-X)

Enables you to specify the name of a file for printing on a local hardcopy device.

For full information on Genera hardcopying, see the section "How to Get Output to a Printer".

Kill or Save Buffers Command

Kill Or Save Buffers (m-X)

Displays a menu listing all existing buffers. Modified buffers are initially marked for saving. Choices are: Save, Kill, Unmodify, and Hardcopy. Specify these options next to the buffer names in the menu. This command is bound to c-X c-m-B and appears on the editor menu. Specifying a numerical argument to c-X c-m-B inhibits the initial marking of the menu.

Hardcopying From Zmail

You can use the following commands when producing hardcopy from Zmail:

- Hardcopy Message Command
- Hardcopy All Command
- Show Printer Status Zmail Command
- Hardcopy File Zmail Command
- Format File Zmail Command

Hardcopy Message Command

Hardcopy Message (m-X)
: Hardcopies the current message. Note that you can also click Right on [Other] in the Zmail menu and select Hardcopy Message. Additionally, you can click Right on the message summary line, and then click Right on [Move] and select Hardcopy.

Hardcopy All Command

Hardcopy All (m-X)
: Hardcopies all the messages in the current sequence. Note that you can also click Right on [Map Over] and select [Hardcopy] for copying all messages in the current sequence.

Note that whether you hardcopy a single message or hardcopy all messages in a sequence, you can click Right on [Hardcopy] and specify the number of copies and which printer to use. The Other option in the list of printers enables you to specify an arbitrary printer, using either the pretty name or the namespace name. This printer becomes the selected printer, and remains in the menu for subsequent hardcopy commands.

Show Printer Status Zmail Command

Show Printer Status (m-X)
Prompts for the name of a printer and displays its print queue.

Hardcopy File Zmail Command

Hardcopy File (m-X)

Sends the file associated with the pathname you specify to the default printing device. The default is the first pathname specified in the File-References: header field. If there is no File-References: field, the default is the current mail file.

Format File Zmail Command

Format File (m-X)

Formats the file associated with the pathname you specify. c-U m-X Format File formats the file and sends it to a printer. The default pathname is the first pathname specified in the File-References: header field. If no File-References: field exists, the default is the current mail file.

Hardcopying from Dired

You can hardcopy files in Dired by marking files. When you exit Dired, the marked files are sent to the printer.

P — Dired Hardcopy File

Marks the current file for printing. Dired puts a P in the first column to show that the file is marked.

You can specify a numeric argument *n*, marking the next *n* files for printing.

Hardcopying the Screen

You can produce a hardcopy of your screen by pressing FUNCTION Q or FUNCTION *n* Q (where *n* is any numeric argument):

Function `Q` — Captures a screen image for hardcopying or inserting in a file.

Function *n* `Q` — (where *n* is any numeric argument, for example `FUNCTION 0 Q`) Displays a menu of options for capturing screen images. Note that the choices you make remain in effect for subsequent uses of `FUNCTION Q`.

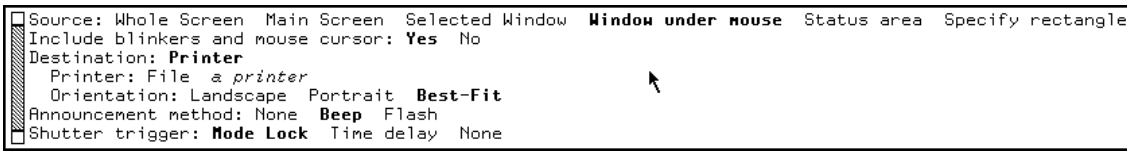


Figure 121. `FUNCTION 0 Q`

`Source:` options identify the area of the screen to capture:

Whole Screen — A bitmap image of the entire screen

Note that a full-screen bitmap image includes a border around the actual screen image. If you do not want this extra whitespace around the image, select the Named Image option and edit the bitmap image using the "Refit Bounding Box Bitmap Editor Command".

Main Screen — All but the status area

Selected Window — The window of the selected activity (for example, Symbolics Concordia)

Window Under the Mouse
The window under the mouse cursor

Use this option when you cannot use the mouse to specify a portion of the screen (for instance, to take a snapshot of a pop-up menu).

Status Area — The window at the bottom of the screen (including the mouse information line(s))

Specify Rectangle — Specify the area of the screen to be captured using the mouse

Window History — The history of the specified window

`Shutter Trigger:`

Mode Lock — Captures the image when the `MODE LOCK` key is pressed. This allows you to *set up* the screen a certain way, perhaps with a menu showing, or the mouse cursor in a certain place.

Time Delay	Allows you to specify a time delay before capturing the image. The default is 5 seconds.
None	Captures the image as soon as you click on [Done]. This is the default.

Hardcopying from the File System Editor

You can produce hardcopy using the system hardcopy menu from FSEdit by following these steps:

1. Click Right on a file name. A menu of file operations appears.

2. Click on Hardcopy in the menu of file operations. The menu in Figure 86 appears.

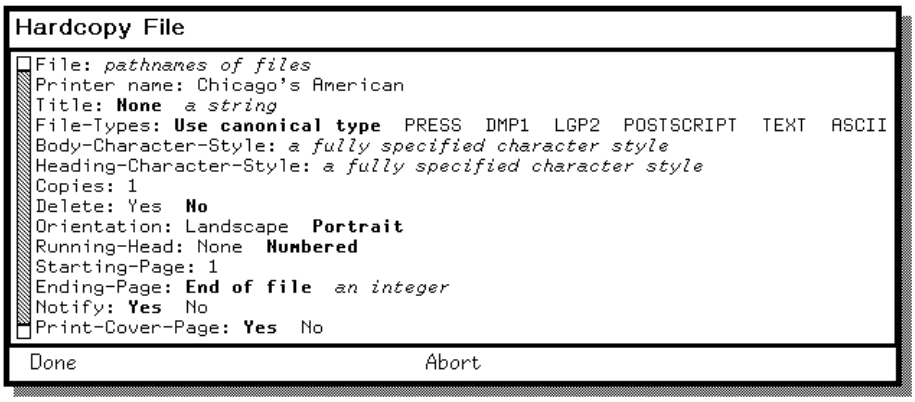


Figure 122. The Hardcopy Menu

3. You can modify any of the parameters displayed. Clicking on Done prints the file.

Changing the default printer

You can change the default printer using the Set Printer Command.

Set Printer **Command**

Set Printer *printer-name keywords*

Sets the default printer for hardcopy.

printer-name	The name of a supported printer that can be reached by your machine.
keywords	:More Processing, :Output Destination, :Output Type
:More Processing	{Default, Yes, No} Controls whether **More** processing at end of page is enabled during the output to interactive streams. The default is Default. If No, output from this command is not subject to **More** processing. If Default, output from this command is subject to the prevailing setting of **More** processing for the window. (See the section "FUNCTION M".) If Yes, output from this command is subject to **More** processing unless it was disabled globally (see the section "FUNCTION M").
:Output Destination	{Buffer, File, Kill Ring, Printer, Stream, Window} Enables you to direct the output of this command. The default is the stream ***standard-output*.**
:Output Type	{text bitmap both} Enables you to specify whether a printer prints text (files and mail messages), bitmap (graphics and screen hardcopy), or both text and bitmap.

Additional Methods of Changing the Default Printer

- You can change the default printer in your init file by specifying the printer most convenient for you. See the function **hardcopy:set-default-text-printer**.

- The Hardcopy File command accepts a keyword argument of :printer enabling you to specify a printer. For example:

```
Hardcopy File q:>kjones>report.pr :printer beacon
```

- Additionally, you can specify a different printer by clicking on the printer name.

- The System menu enables you to specify a different printer by clicking on the printer name.

You can view the default printer using the Show Printer Defaults command.

Show Printer Defaults **Command**

Show Printer Defaults *keywords*

Displays the current default printer(s). If you send all your hardcopy output to one printer, the command returns:

```
Default Printer (for both text and bitmap output): printer-name
```

If you use a different printer for text and screen hardcopy, the command returns:

```
Default Text Printer: printer-name1
Default Bitmap Printer: printer-name2
```

Printer Error Messages

LGP2 and LGP3 Printer Error Messages

For an LGP2/LGP3 printer named Pravda, the print spooler (if the print spooler software is installed) or the printer might send the following error messages. Below each error message is an explanation of the error.

LGP2 and LGP3 Irrecoverable Error Messages

If you receive an error message that begins with:

```
Irrecoverable error on printer "Pravda":
```

this indicates an irrecoverable problem, meaning that you must requeue any print requests that are pending at the time the error message appears. Irrecoverable error messages can be of two types for the LGP2 and LGP3:

```
Irrecoverable error on printer "Pravda":  irrecoverable device error
```

This indicates that the LGP2/LGP3 rejected the commands sent from the Symbolics computer. This might indicate an error in the PostScript code sent by the Symbolics computer, or it might indicate a noisy serial line.

```
Irrecoverable error on printer "Pravda":  Pravda, a LGP2 printer
supposedly connected via SERIAL interface, does not appear to be
connected.
```

This indicates that the cables running from the Symbolics computer to the printer are not properly installed. You need to check the serial cable connection when this error appears.

LGP2 and LGP3 Recoverable Error Messages

If you receive an error message that begins with

```
Printer "Pravda" needs intervention:
```

this indicates a recoverable problem, meaning that you can repair the problem and continue your print requests. Once you have fixed a recoverable error message, this appears:

```
Printer "Pravda" is operating again
```

There are four types of recoverable error messages for the LGP2 and LGP3 printers:

```
Printer "Pravda" needs intervention:  Out of paper
```

This indicates that the printer needs more paper. Fill the paper tray to fix this problem.

```
Printer "Pravda" needs intervention:  No paper tray
```

This indicates that the paper tray is not inserted into the printer. Place the paper tray into the printer to fix this problem.

```
Printer "Pravda" needs intervention:  Jam
```

This indicates a paper jam. See the LGP2 or LGP3 manuals for recommendations on how to fix this problem.

```
Printer "Pravda" needs intervention:  Cover open
```

This indicates that the printer cover is open. To fix this problem, push the cover down until it clicks into place.

ASCII Printer Error Messages

```
>>Error:  Mondrian, an ASCII printer supposedly connected via SERIAL interface,
does not appear to be connected.
```

This appears if you are trying to print a job on your ASCII printer and the printer is not asserting DTR. This message occurs only if the print server does not have the print spooler software installed.

```
[08:53:28 ASCII printer "Mondrian" doesn't appear to be connected to its
SERIAL interface; it will not be spooled.]
```

This appears in the print log if you are trying to print a job on your ASCII printer, and the ASCII printer is not asserting DTR. This message occurs only if the print server has the print spooler software installed.

The above error *can* appear even if your printer *is* asserting DTR. In this case, someone might have disconnected the ASCII printer cable.

DMP1 Printer Error Messages

For a DMP1 printer named Senegal, the Print Spooler (if the Print Spooler software is installed) or the printer may send the following error messages. Below each error message is an explanation of the error.

DMP1 Irrecoverable Error Messages

If you receive an error message that begins with:

```
Irrecoverable error on printer "Senegal":
```

this indicates an irrecoverable problem, meaning that you must requeue any print requests pending at the the error message appears.

Here is a specific irrecoverable error message for the DMP1:

- Irrecoverable error on printer "Senegal": Serial connection lost

  This error message indicates that the cables running from the Symbolics computer to the printer are not properly installed. You need to check the serial cable connection if this error appears.

DMP1 Recoverable Error Messages

If you receive an error message that begins with

```
Printer "Senegal" needs intervention:
```

then this indicates a problem which is recoverable, meaning that you can replace whatever is missing without requeuing your print requests. Once you have fixed a recoverable error message, the message *Device Pravda is operating* appears. Recoverable error messages of can be of two types for the DMP1:

```
Printer "Senegal" needs intervention:  Paper out
```

This error message indicates that the printer is out of paper. To fix the problem feed new paper into the printer and press the **ON-LINE** button.

```
Printer "Senegal" needs intervention:  Cover open
```

This error message indicates that the cover is open. To fix the problem, put the cover down, and press the **ON-LINE** button.

SCSI Devices for XL Machines

Installing a SCSI Device

This section describes how to physically connect a SCSI device to the XL machine, how to configure the device with an appropriate SCSI address.

Connecting the Device

1. Find an appropriate cable. You need a cable that has a standard SCSI connector (a 50-pin D connector) on each end. Note that you cannot use the cable provided by Storage Dimensions with the MacinStor SCSI disk, because one end of that cable does not have a standard SCSI connector.

2. Connect the SCSI cable to the SCSI connector in the back of the optical disk drive. Connect the other end of the SCSI cable to the SCSI connector in the back of the XL machine.

3. Be sure that the SCSI bus is terminated properly. Whatever SCSI device is at the end of the SCSI bus must have a SCSI terminator on it. If you need more information about how to connect SCSI devices, see the section "Attaching a SCSI Device".

Choosing the SCSI Address for the Device

1. Choose an unused SCSI address for the device.

   Each SCSI device on the SCSI bus must have a unique SCSI address between 0 and 7 (inclusive). SCSI address 0 is used by the Ivory processor board, so that is unavailable.

   If you have other SCSI devices attached, you need to find out what SCSI addresses they use. The Show Machine Configuration command gives the SCSI addresses of any devices attatched to your XL machine.

2. When you have chosen a SCSI address, configure the device by setting theSCSI address according to the device manufacturer's instructions.

On XL machines, Genera supports the Storage Dimensions MacinStor Erasable Optical 1000 disk drive. You can create a FEP file system on an optical disk, and access the FEP file system in the usual ways. You can use the optical disk to store large files such as image files, world loads, and Statice databases. (You cannot boot locally off a world load stored on an optical disk, but you can store the world load there for archival purposes and for netbooting from other machines.)

Cartridges formatted for use on XL machines are not compatible with MacIvory and vice-versa. On MacIvory, the MacinStor is treated like any other disk drive. MacIvory cartridges contain a standard Macintosh file system, and within that file system you may create one or more Ivory partitions. When using a MacinStor on a MacIvory, the procedures and commands described in the following sections do not apply. See the section "Using the MacIvory Control Panel".

Normal Use of an Optical Disk

The initial installation instructions are described in the section "Installing a SCSI Device". Once the disk drive is installed, and an optical disk has been initialized with a FEP filesystem, the normal use is as follows:

1. Insert the optical disk into the drive.

2. Mount the optical disk using the Mount Optical Disk CP command.

3. Use the optical disk by giving Genera commands in the same way that you would access any FEP filesystem. You can write files to the FEP filesystem and read them back. You can do Show FEP Directory on the FEP filesystem. (Note the restrictions below on using the optical disk for paging, or with LMFS, or with worlds.)

4. Dismount the optical disk using the Dismount Optical Disk CP command.

5. Eject the optical disk from the drive by pressing the button on the front of the drive.

Guidelines for Use of Optical Disks

It is important to understand that you cannot dismount the optical disk while any application is accessing it (for example, when there is an open stream to the optical disk). Also, you cannot eject the optical disk until it has been dismounted. The software prevents you from doing these things, in order to protect the integrity of the data on the FEP filesystem on the optical disk. This has the following implications:

Using the optical disk for paging is not recommended.
Once you have added a paging file that is stored on an optical disk, you cannot dismount (or eject) the disk until you next cold-boot Genera. It might be useful in an emergency to set up a paging file on the optical disk and use it long enough to save your files, knowing that you will cold boot soon anyway.

Putting a LMFS on an optical disk is not recommended.
Once you have activated a LMFS that is stored on an optical disk (which happens the first time you try to access any file from that LMFS), you cannot dismount (or eject) the disk until you next cold-boot Genera. This is because LMFS has no concept of a dismountable disk, and the integrity of the data must be protected by preventing the disk from being dismounted while LMFS is active. It might be useful in some emergency situations to create a single-partition LMFS on an optical disk and use it briefly, knowing that you will cold boot soon anyway. On the other hand, you can store any file in the FEP filesystem of the optical disk, so there seems little point in creating a LMFS on the optical disk.

Do not write protect erasable optical disks that you use as FEP filesystems. The software for accessing the FEP from Lisp cannot read from write-protected disks.

There is an additional restriction:

Using the optical disk for booting a world is not possible.
The optical disk drive is not recognized by the FEP until Genera is up and the Mount Optical Disk command has been given. Therefore, you cannot boot from a world stored on an optical disk. You can, however, store worlds on an optical disk for archival purposes or to use the optical disk host as a Netboot server.

Setting up a Blank Optical Cartridge

1. When preparing to use a blank optical cartridge, insert it into the drive. Note that each cartridge has two sides which must be prepared separately. Only one side is accessible at any time. In Genera, the first thing to do is give the Write Partition Map CP command, and give the appropriate SCSI address as an argument. There are keyword arguments as well, but the defaults are usually appropropriate.

   `Write Partition Map` *SCSI-address*

   Note that this command will erase any data on the disk, and create a new partition map on it.

2. Mount the optical disk by using the Mount Optical Disk CP command.

   `Mount Optical Disk` *SCSI-address*

   This command makes the optical disk accessible to Genera.

3. Create a FEP filesystem by giving the Create Initial FEP Filesystem command.

   `Create Initial FEP Filesystem` *FEP-unit-number*

   Note: Create Initial FEP Filesystem *differs* from the other commands used here in that it takes a FEP unit number, *not* a SCSI address.

   The convention is that the FEP unit number is 7 greater than the SCSI address. For example, if you set up the optical disk drive at SCSI address 4, the FEP unit is FEP 11. (Note that if you have multiple I/O boards, and the optical disk drive is attached to the second I/O board, then the FEP unit is 23 plus the SCSI address.)

   Now the optical disk has a FEP Filesystem on it, which you can access from Genera. Once the FEP filesystem has been created, you refer to it by the FEP unit number, not the SCSI address.

4. Show the FEP directory, as a test, by using the Genera CP Command:

   `Command: Show FEP Directory`

   You should see the new FEP unit appear along with the other FEP units.

Commands for Using Optical Disks

Write Partition Map Command

Write Partition Map *SCSI-address keywords*

Writes a new partition map on an optical disk. This command erases any data or partitions already on the disk.

SCSI-address The SCSI address of the optical disk drive.

keywords :Partition Map Size, :Apple Driver Size, :Apple Hfs Size, :FEPFS Size

The *keywords* enable you to control the size of four partitions. The values are in number of blocks, or Rest. If you supply keywords, three of the partitions should be specified as integers, and one should be Rest (indicating that the remaining space should be used for that partition).

:Partition Map Size
The default is 40 blocks.

:Apple Driver Size
The default is 30 blocks.

:Apple Hfs Size The default is 0 blocks.

:FEPFS Size The default is Rest.

Create Initial FEP Filesystem Command

Create Initial FEP Filesystem *FEP-unit*

Creates an initial FEP filesystem on the given *FEP-unit*. This command is used for initializing an optical disk or a SCSI disk.

FEP-unit The number of the FEP unit.

The convention for mapping SCSI addresses to FEP unit number is that the FEP unit number is 7 greater than the SCSI address. For example, if you set up the SCSI disk drive at SCSI address 4, the FEP unit is FEP 11. (Note that if you have multiple I/O boards, and the SCSI disk drive is attached to the second I/O board, then the FEP unit is 23 plus the SCSI address.)

Mount Optical Disk Command

Mount Optical Disk *SCSI-address*

Mounts the optical disk indicated by *SCSI-address*. Once the disk is mounted, if it has a FEP filesystem on it, you can access it by using its FEP unit number. Otherwise, you should use the Create Initial FEP Filesystem command.

SCSI-address The SCSI address of the optical disk drive.

The convention for mapping SCSI addresses to FEP unit number is that the FEP unit number is 7 greater than the SCSI address. For example, if you set up the optical disk drive at SCSI address 4, the FEP unit is FEP 11. (Note that if you have multiple I/O boards, and the optical disk drive is attached to the second I/O board, then the FEP unit is 23 plus the SCSI address.)

Dismount Optical Disk Command

Dismount Optical Disk *SCSI-address keywords*

Dismounts the optical disk indicated by *SCSI-address*. Once the disk is dismounted, you cannot access it again until you mount it.

SCSI-address	The SCSI address of the optical disk drive.
keywords	:Eject
:Eject	{Yes, No} Whether to eject the optical disk after it is dismounted. The default is No. The mentioned default is Yes.

On XL machines, Genera 8.1 supports the Storage Dimensions MacinStor SCSI disk drive series. Currently, only the MacinStor 320 and the MacinStor 650 have been qualified for use with XL machines. However, other SCSI disk models may also work properly.

Setting up a Blank SCSI Magnetic Disk

Note that new SCSI disks cannot be initialized from the FEP. You need an XL running Lisp to set up a new disk.

1. Connect the SCSI disk to the XL and power up the disk. Use the Format SCSI Disk command to format the disk. Provide the appropriate SCSI address as an argument.

   ```
   Format SCSI Disk SCSI-address :Sector Size 1280
   ```

   Note that this command will erase any data on the disk.

2. Warm boot Lisp so Genera will recognize the formatted drive.

3. Create a FEP filesystem by giving the Create Initial FEP Filesystem command. Provide the appropriate FEP unit as an argument.

   ```
   Create Initial FEP Filesystem FEP-unit
   ```

   The convention for mapping SCSI addresses to FEP unit numbers is that the FEP unit number is 7 greater than the SCSI address. For example, if you set up the SCSI disk drive at SCSI address 4, the FEP unit is FEP 11. (Note that if you have multiple I/O boards, and the SCSI disk drive is attached to the second I/O board, then the FEP unit is 23 plus the SCSI address.)

   Now the SCSI disk has a FEP Filesystem on it, which you can access from Genera. Once the FEP filesystem has been created, you refer to the disk by the FEP unit number, not the SCSI address.

4. Show the FEP directory, as a test, by using the Genera CP Command:

```
Command: Show FEP Directory
```

You should see the new FEP unit appear along with the other FEP units.

Commands for Using SCSI Magnetic Disks

Format SCSI Disk Command

Format SCSI Disk *SCSI-address*

Formats the SCSI disk indicated by *SCSI-address*.

SCSI-address	The SCSI address of the SCSI disk drive.
keywords	:Controller, :Sector Size
:Controller	No documentation supplied
:Sector Size	The default is 1280 bytes

Create Initial FEP Filesystem Command

Create Initial FEP Filesystem *FEP-unit*

Creates an initial FEP filesystem on the given *FEP-unit*. This command is used for initializing an optical disk or a SCSI disk.

FEP-unit	The number of the FEP unit.

The convention for mapping SCSI addresses to FEP unit number is that the FEP unit number is 7 greater than the SCSI address. For example, if you set up the SCSI disk drive at SCSI address 4, the FEP unit is FEP 11. (Note that if you have multiple I/O boards, and the SCSI disk drive is attached to the second I/O board, then the FEP unit is 23 plus the SCSI address.)

Using SCSI Tape Drives

SCSIQIC-11 tapes and 6250 bpi tape drives now work on XL machines. The initial installation instructions are described in the section "Installing a SCSI Device". Once the tape drive is installed, you can access it in the same way you access other supported tape devices. Note that if your XL is already running Lisp when you connect the new tape drive, you have to warm boot before the new tape drive is recognized.

Some XL and MacIvory machines use the Emulex MT02 controller, a QIC-11 SCSI tape drive. You need to add a `PERIPHERAL` entry to the namespace object of any host that uses the Emulex MT02 controller. If this entry is not present, only four of

the nine tracks will be used. For example, for a controller at SCSI address 1, the entry should look like this:

```
Peripheral: TAPE UNIT SCSI1 MODEL EMULEX-MT02
```

Statice Runtime

Overview of Statice Runtime

Statice is an object-oriented database system for the Genera programming environment. Statice provides client programs with *persistent*, *shared* storage of information. Persistent information stored in Statice exists outside and beyond the boundaries of the Lisp world that created it, and is protected against failure. Shared information is shared by distinct Lisp worlds on different workstations, for writing as well as reading.

The tools to develop Statice applications are available in the Statice Developer product, which is a layered product. The Statice Developer product includes documentation of how to develop Statice applications.

Statice Runtime is a separately loadable system. To load Static Runtime, type :Load System Statice-Runtime. Statice Runtime enables Genera users to use, access, and maintain a Statice database. The benefit of making Statice Runtime available to Genera users is to make it easier and more practical to deliver Statice-based applications; customers of Statice-based applications do not need to buy Statice Developer.

Statice Runtime is a subset of Statice Developer. The functions and macros which enable programmers to develop Statice applications are not present in Statice Runtime. Statice Runtime is documented here.

Operations and Maintenance of Statice Databases

The Architecture of Statice

Using Statice Locally or Remotely

Statice can be used on a single host, or between many hosts across the network. This section describes how this works, and explains the terminology used for the participants and the roles they play.

A *host* is a computer, a workstation. A *site* is a collection of hosts (and other things, such as users), all at more or less the same physical place. Every host is at one particular site. We assume that every host at a site is connected to a network, so that each host can communicate with every other host. Hosts, sites, and networks are all described by the namespace database. For more information about the namespace database and the things it describes: See the section "Concepts of

Symbolics Networks". See the section "Setting Up and Maintaining the Namespace Database".

A *Statice File System* is a file system that holds Statice databases. Every Statice File System is at one particular site. Every Statice File System resides on one particular host at that site. Each Statice File System is described in the namespace database by a File System namespace object: See the section "How a Statice File System is Described in the Namespace". See the section "Attributes for Objects of Type "File System"".

Suppose that at some site there are two hosts named Mars and Venus, and there are two Statice File Systems named Rose and Iris. Rose resides on host Mars, and Iris resides on host Venus. Fig. ! shows hosts represented as rectangles, and Statice File Systems represented as circles.

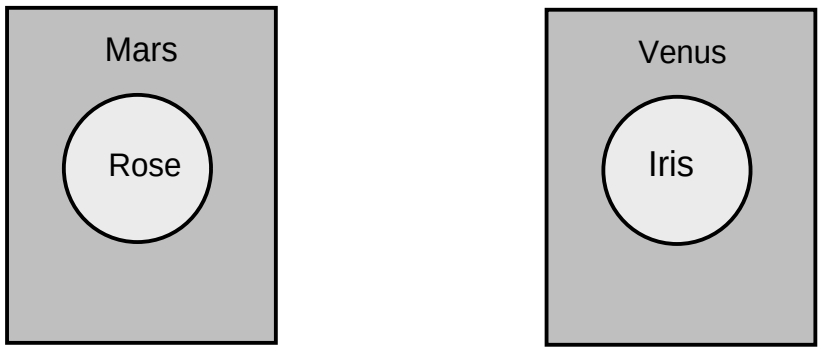


Figure 123. Hosts Mars and Venus, with File Systems Rose and Iris

Now, suppose a process running on host Mars begins using Statice, doing **statice:with-database** and **statice:with-transaction**, calling accessor functions, and so forth. This process might be a Dynamic Lisp Listener, a process associated with some program defined by **dw:define-program-framework**, or any process at all. A process that calls Statice functions and special forms is a *client* process.

If the client process uses a database that resides in the Statice File System named Rose, Statice notices that Rose is on the same host as the client process itself. We say that the client process is using Statice *locally*, or that the client is accessing a *local database*. When a client is using Statice locally, the client manipulates the database directly, invoking the disk driver to access the host's disks, etc.

If the client process uses a database that resides in the Statice File System named Iris, Statice notices that the Iris is on some other host than the client process. We say that the client process is using Statice *remotely*, or that the client process is accessing a *remote database*. When a client uses Statice remotely, a *server process* is created on the *remote host*, and a network connection is created to allow the client process and the server process to communicate. The client process cannot directly access the disks of another host, and so it delegates this work to the server process.

If a second client process, running on host Venus, accesses the same database on

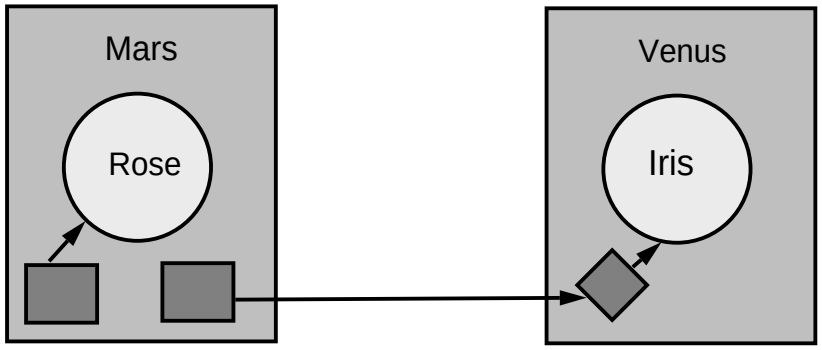


Figure 124. Local and Remote Use of Statice

Iris, this second client is using Statice locally. In Fig 124, we have two client processes, both using the same database, one locally and one remotely. In general, there might be any number of client processes accessing a database, many locally and many remotely.

How a Statice File System is Described in the Namespace

Every Statice File System is described by an object in the namespace database. For more information about the namespace database and the things it describes: See the section "Concepts of Symbolics Networks". See the section "Setting Up and Maintaining the Namespace Database".

The namespace object for a Statice File System is of type File-System (as opposed to Host, Network, User, et. al.), and would typically look similar to this:

```
Host: CHICOPEE
Type: DBFS
Root Directory: FEP1:>Statice>iris.UFD
Pretty Name: IRIS
User Property: PARTITION0 FEP1:>Statice>iris-part0.file.newest
User Property: PARTITION1 FEP1:>Statice>iris-part1.file.newest
User Property: PARTITION2 FEP1:>Statice>iris-part2.file.newest
User Property: LOG-DESCRIPTOR-FILE-ID 1015371-1311996-1048993
User Property: DBFS-DIR-ROOT-FILE-ID 1014172-1311996-1048993
```

When you use the Create Statice File System command, Statice automatically creates a file-system object. See the section "Create Statice File System Command".

The most important attribute is the Host, which says where this file system lives. In this example, Iris lives on host Mars.

The Type field always has the value DBFS. Any other values for this field are reserved for future expansion.

The Root Directory field contains a pathname of a FEP FS file. That file contains the directory of the Statice File System. The pathname should always start with FEP*n*: and end with the .UFD file extension.

The Pretty Name and Short Name mean the same thing as they do in other namespace objects. The Short Name can be used on input as an abbreviation; in particular, you can use it in pathnames. The pretty name is used to display the name of the file system.

The User Properties named PARTITION have values that are pathnames of the FEP files which are the partitions that make up the file system. The database is stored in a number of partitions. For more information on partitions: See the section "Create Statice File System Command".

The User Property named LOG-DESCRIPTOR-FILE-ID is the unique ID of Statice's "log descriptor" file, an internal file used to store various per-file-system information.

The User Property named DBFS-DIR-ROOT-FILE-ID contains, in the form of a string, the internal unique ID of the special database in the file system that stores the hierarchical directory structure of the file system. This is established when the Statice File System is created, and you should never change it.

For reference documentation on the File-System object and its attributes: See the section "Attributes for Objects of Type "File System"".

Why is there a separate File-System namespace object? Why doesn't Statice use the host object, as LMFS does? There are three reasons:

- Pathnames starting with "MARS:" already mean "the LMFS found on host MARS". They can't also mean a Statice File System. We must have a different name in order to specify the Statice File System, because the name "MARS" has already been taken. This is the most compelling reason.

- It's possible to have more than one Statice File System on the same host, each with its own name, using different areas of the host's disks. Such a configuration might be desirable for various administrative reasons.

- You can move a Statice File System from one host to another, using removable disk packs, magnetic tape copies, or moving disks. If you do this, you need to change the file-system namespace object for the Statice File System to refer to the new host. All software that refers to the Statice File System by name continues to work, because the file-system namespace object provides the link to the new location of the Statice File System.

Statice Database Pathnames

Many Statice commands take a pathname as an argument, indicating a database. This should be a *database pathname*. The most striking difference between a database pathname and an ordinary pathname is the first component: for a database pathname, this is a Statice File System; for an ordinary pathname, this is a host.

Every Statice File System contains a set of files. Each file contains a Statice database, and is named by a database pathname.

Database pathnames resemble LMFS pathnames. The "host name" part is the name of the Statice File System, followed by a colon. After that are the names of the directories, separated by greater-than characters. The last part of the pathname is the name of the file. For example, the pathname **Iris:>george>financial>ledger** names a file in the Statice File System named Iris. The root directory on Iris contains a directory named **george**, which in turn contains a directory named **financial**, which in turn contains a file named **ledger**.

There are some important differences between database pathnames and LMFS pathnames. Database pathnames have a name, but no type or version. Statice File System directories store fewer properties than LMFS directories; there is no modification date, reference date, do-not-reap flag, and so on. There are also no links, only files and directories. Statice File System directories support the following properties, which can be examined with directory listings (see **fs:directory-list** or **fs:file-properties**).

:author	The user ID of the creator of the file.
:creation-date	The date and time at which the file was created, expressed as a universal time.
:comment	An arbitrary string that appears in directory listings.
:directory	**t** if this is a directory, **nil** if this is a file.
:length-in-blocks	The length of the file, in blocks. A block is 1152 (decimal) bytes, the size of a FEP file system block. (FEP blocks are 1152 bytes on 3600-family machines, and 1280 bytes on Ivory machines.)

The **:author**, **:creation-date**, and **:comment** properties can be set using **fs:change-file-properties**. **:comment** can be set for files but not directories. The other properties cannot be set.

Database pathnames are case-insensitive for lookup, like those of LMFS. When you make a new file, the Statice File System directory remembers the case you used and stores this in the directory. To look up a file that already exists, you can use either case for any character. A directory cannot have one file named **Foo** and another named **foo**; these are considered to be the same name.

Database pathnames support relative pathname syntax, wildcard syntax, and completion, just like LMFS pathnames. **<foo>bar** means the file named bar in the directory named foo in the directory that is the parent of the default pathname's directory. **>foo>*** means all the files in the directory named foo. **>foo>**>*** means all the files in the directory named foo and its descendants. **>foo>b*** means all the files whose names start with **b** in the directory named **foo**.

There is no undeletion, and no expunging. When you delete a file, it is immediately and permanently removed.

Files cannot be opened by the Lisp **open** function because they are not really files in the sense of the Lisp stream system. The contents of files are accessed only via Statice operations, never by streams. The only exception is if they are opened with a **:direction** of **:probe**, **:probe-directory**, or **:probe-link**. This lets programs use **probe-file** to check for the existence of files in a Statice File System.

Database pathnames work correctly with the Genera directory manipulation tools, such as DIRED, FSEdit (the File System Editor), and commands such as Show Directory, Create Directory, Delete File, and Rename File. However, they do not work with commands that attempt to open files, such as Copy File and Show File.

The root directory of every Statice File System directory contains an entry named **Directory**, which refers to the directory itself. This gives you a way to name the root directory as a file, in order to perform operations on it. This is rarely necessary, and you can usually just ignore the **Directory** entry.

Dealing with Databases by Their Pathnames

Many operations on databases can be done by using normal file commands on the database pathname. See the section "Statice Database Pathnames".

- What databases are stored in a directory of a given Statice file system? Use the Show Directory command:

  ```
  Show Directory beet:>fred>*
  ```

- What are all the databases stored an entire Statice file system? Use the Show Directory command:

  ```
  Show Directory beet:>**>*
  ```

- How can I rename a database? Use the Rename File command.

  ```
  Rename File beet:>university beet:>harvard
  ```

- How can I remove a database? Use the Delete File command on a database pathname.

  ```
  Delete File beet:>university
  ```

  Although this method permanently removes the file, you can restore the file from backup tapes (if you have any). See the section "Selective Restore Command".

Services and Protocols Used by Statice

Statice uses several network services and protocols. The commands that install Statice at your site add new information to your namespace host objects, indicating that these services and protocols are supported. In this section, we briefly describe the new namespace information; this information is not required in order to write

Statice programs, but it might be useful to help you debug any unusual name-space-related problems.

DBFS-PAGE Service

DBFS-PAGE is the network service provided by a Statice server for the benefit of Statice clients. When a Statice client first accesses a particular Statice server host, it invokes the DBFS-PAGE service on that host. From then on, this client uses this connection to communicate with that host, even if it accesses more than one Statice File System on that host.

When the transaction ends, the connection is returned to a free pool, so subsequent transactions can use that connection. This helps reduce the amount of time spent opening network connections.

Statice servers use the same connection scavenger mechanism used by Genera file servers, so that if a connection hasn't been used for a long time, the server process is killed and the connection goes away. See the section "The File Control Lifetime Host Attribute".

The namespace entries for the DBFS-PAGE service should be present in the host object of every host used as a Statice File System server. The Add DBFS Page Service command sets up this service entry in the namespace database. See the section "Add DBFS PAGE Service Command".

See the section "Using Statice for the First Time".

DBFS-Page Protocol

DBFS-PAGE is the name of a network protocol that implements the DBFS-PAGE service. This protocol is built on top of the byte-stream-with-mark network medium. See the section "BYTE-STREAM-WITH-MARK Network Medium". It can be used with Chaosnet or TCP/IP networks. For Chaosnet, the contact name is "DBFS-PAGE"; for TCP/IP, the port number is 569.

ASYNCH-DBFS-PAGE Service

ASYNCH-DBFS-PAGE works in the reverse direction: the Statice server uses this service to form a connection back to Statice File System client hosts. The server uses this connection to notify the user when pages have been modified by another client; the client acts on this information by invalidating its cache.

The service that ASYNCH-DBFS-PAGE provides is not necessary for correct operation of Statice. If the server finds that it cannot form a connection to the client, it simply gives up and tries again later. The cache coherency protocol within Statice makes sure that invalid data is never used. However, Statice will be more efficient if ASYNCH-DBFS-PAGE is working properly. You should try to make sure that all hosts that use Statice as a client have the proper namespace entries for ASYNCH-DBFS-PAGE in the namespace database. The command Add ASYNCH DBFS PAGE Service should be run on each client host: See the section "Using Statice for the First Time".

ASYNCH-DBFS-PAGE Protocol

ASYNCH-DBFS-PAGE is the name of the network protocol that implements the ASYNCH-DBFS-PAGE service. This protocol is built on top of the byte-stream-with-mark network medium. See the section "BYTE-STREAM-WITH-MARK Network Medium". It can be used with Chaosnet or TCP/IP networks. For Chaosnet, the contact name is "ASYNCH-DBFS-PAGE"; for TCP/IP, the port number is 568.

Attributes for Objects of Type "File System"

Host

Specifies the host that the file system resides on; a host object (required).

```
Host: MARS
```

Type

Must always be DBFS (required). Other values are reserved for future expansion.

```
Type: DBFS
```

Root Directory

Specifies a pathname of a FEPFS file. That file contains the directory of the Statice File System. The pathname should always start with FEP*n*: and end with the .UFD file extension.

```
Root Directory: FEP1:>Iris.UFD
```

Pretty Name

Specifies a name for the file-system to use when showing the name; a token (required).

```
Pretty Name: Iris
```

Nickname

Specifies alternate names for the network; a set of names. The file system may be found by these names.

```
Nickname: IRE
```

Short Name

Specifies additional nicknames; a set of names. A short-name is used when a program wants to display a host's name without using up too much space. A short-name is used for both input and output. This is also used in the printed representation of pathnames.

```
Short Name: I
```

User Property

User-Property

All objects contained within the namespace (hosts, sites, namespaces, printers, and users) are eligible to have a User-Property attribute. It consists of a pair whose first element is an indicator (like that of a property list) and whose second element is a token. The User-Property attribute holds any information that users choose to associate with an object. For example:

```
User-Property: ID-number 123-45-6789
```

Statice automatically places several user properties into file-system objects. The User Properties named PARTITION have values that are pathnames of the FEP files which are the partitions that make up the file system. The database is stored in a number of partitions. For more information on partitions: See the section "Create Statice File System Command".

The User Property named LOG-DESCRIPTOR-FILE-ID is the unique ID of Statice's "log descriptor" file, an internal file used to store various per-file-system information.

The User Property named DBFS-DIR-ROOT-FILE-ID contains, in the form of a string, the internal unique ID of the special database in the file system that stores the hierarchical directory structure of the file system. This is established when the Statice File System is created, and you should never change it.

FEP File for Generating Statice Unique IDs

When you first run Statice on a machine, it automatically creates a small file (2 blocks) in the FEP file system, whose name is:

FEP*n*:>UNIQUE-ID.FEP.1

where *n* is the lowest fixed-medium disk unit, or the lowest disk unit if all units are removable-medium. (In almost all configurations, *n* is zero.)

This file is used internally by Statice to generate unique IDs. We recommend that you leave it there, and don't delete it. If you do delete it, Statice will re-create it next time Statice is run.

Statice File System Operations Program

The Statice File System Operations program is an interactive utility for maintaining and manipulating Statice File Systems. It serves primarily as the user interface to the backup system. It also provides commands for enabling and shutting down a Statice File System, and other functions.

We first present several sections that give background information, and then describe how to use the program itself, in the section "Using the Statice File System Operations Program".

Some operations on databases can be done by using normal file commands on the database pathname. For details: See the section "Dealing with Databases by Their Pathnames".

Overview of the Statice Backup Facilities

The Statice File System backup facilities let you make backup copies of the databases stored in a Statice File System, onto another medium. If the disk holding the Statice File System is damaged or destroyed, the copy of the Statice File System can be restored onto a fresh disk.

It is very important to back up your Statice File System on a regular basis. Disks are fragile and subject to failure. Doing backups is the only way to protect your data against disk failures.

The backup facilities copy database information to tertiary storage media. Currently, industry-standard magnetic tape and cartridge tapes are supported. The software names and catalogs the media into groups called volume sets.

Two forms of backup are provided: complete backups, and continuous archive backup. (Currently only complete backup is supported.) Complete backup makes a complete copy of a database file system onto tertiary storage, and assures the copy is transaction-consistent. If the database file system is destroyed, you can restore the copy, losing only changes made since the latest backup copy was produced. Archive logging continuously copies all database changes to tertiary storage. If the database file system is destroyed, you can restore the latest complete backup copy, and then replay the archive changes, so that no information is lost.

A backup tape from a Statice File System stored on a 3600-family Statice server cannot be reloaded into an Ivory Statice File System, and vice versa. Also, if you want to move whole databases between file systems of different block size, you have to use the high-level dumper (the Dump Database and Load Database commands) to convert the data into a text file, and then move the text file.

See the section "High-level Dumper/Loader of Statice Databases".

Kinds of Tertiary Storage

Backup copies are kept on *tertiary* storage. (Primary storage is main memory, and secondary storage is disk.) The Statice File System backup facilities back up to and restore using a *generic tertiary storage protocol*, so that different kinds of tertiary storage can be used interchangably, and new kinds can be added in the future.

With Symbolics 36xx systems, two kinds of tertiary storage are currently supported: 1/4-inch cartridge tapes, and 1/2-inch industry-standard reel-to-reel magnetic tapes. (The fraction of an inch refers to the width of the tape itself.) Both kinds of tapes can be used either locally or remotely. In local usage, the tape drive hardware is physically connected to the workstation doing the backup or restore. In remote usage, the tape drive hardware is connected to some other computer, which communicates with the workstation over the network.

In future releases, we we plan to support write-once optical disks as another tertiary storage medium. We also intend to support other formats of magnetic tape as hardware support becomes available.

The physical integrity of backup copies is very important. If it's necessary to restore a file system from a backup copy, it is imperative that the data be readable from the copy. Unfortunately, magnetic tapes are an imperfect physical medium. Periodically, tapes are found to be unreadable, or to contain data errors.

To protect against problems with tapes, the backup system always writes data using a powerful error-correcting coding technique, a simple version of Youngquist's algorithm. The technique is effective at recovering from large damage spots on tape; this is important, since it is not uncommon for 100 sequential bits to "drop out". The cost of the algorithm is that approximately 3/2 as much tape is required to store the same amount of information. We believe that the protection is worth the cost of the extra tape. We tested this coding technique by scratching a tape with a razor, and the data were still recovered.

If you have a choice between industry-standard and cartridge tape, industry-standard tape is preferable, because:

- Industry-standard tape runs faster than cartridge tape.
- More data fits on a single industry-standard tape than on a cartridge tape, so you don't have to change tapes as often.

The primary advantage of cartridge tape is its lower cost. Every Symbolics site has

at least one cartridge tape drive, because cartridge tape is used for distributing software.

The benefits of using an industry-standard tape drive increase if you have large Statice File Systems, or if you write new data frequently.

Choosing the Kind of Tertiary Storage to Use

If you have a choice between local and remote usage, local usage is preferable, because:

- Network connections are inherently unreliable. It is inconvenient to have a network connection fail during a backup or restore operation.
- Local usage runs faster than remote usage.

Therefore, we recommend that you put your Statice File System on a workstation that has its own tape drive, if possible.

When a command of the Statice File System Operations program prompts for a "device specification", it wants to know which tape drive to use. (The prompt doesn't say "tape drive" because there will be other kinds of tertiary storage in the future.) The default is to use the cartridge tape drive on the local workstation. You can change the host, to use a tape drive on another computer, and you can change the device type to industry-standard tape.

If you select industry-standard tape, the prompt expands to let you specify a unit number and a density. The unit number parameter is provided because it is possible to attach more than one industry standard tape drive to a computer; the unit number distinguishes which one you want. The default value is zero, and if there is only one tape drive, it should be unit zero.

Industry standard tapes can be written at several different densities, measured in bits per inch. We support 1600, 3200, and 6250 bits per inch, defaulting to 3200. Not every tape drive can read and write at every density! You must check the hardware you intend to use, and determine what densities it supports.

Volume Capacity

When you make a backup copy, you don't need to know in advance how many volumes will be needed to hold the copy. Whenever the dumper reaches the end of one volume, it asks you to mount another volume, until the copy is completed.

However, you might want to be able to estimate, in advance, how many tapes will be needed to hold a copy. Here is some information to help you make such an estimate. (There is no requirement that you make such an estimate, but it is sometimes desirable.)

It's difficult to make an accurate estimate of how many volumes are needed for a backup copy, because the number depends on many factors. More volumes are needed if there are many small files than one large file. Different tape drives have

different characteristics; industry-standard tape drives don't all write interrecord gaps the same way, and cartridge tape drives are affected by the quality of the tape medium and the cleanliness of the heads. So the following numbers should be treated as approximate figures.

The numbers below show the actual data capacity for several different kinds of tape. The actual data capacity is smaller than the raw capacity primarily because of the overhead of the error-correcting coding used in Statice File System backup tapes. Capacities are given in megabytes.

DC300XL/P cartridge tape: 30 MB

DC600A or DC600XTD cartridge tape: 40 MB

600-foot industry standard tape at 3200 bpi: 14 MB

2400-foot industry standard tape at 3200 bpi: 60 MB

Industry-standard tapes can be written at different densities; the numbers above assume a density of 3200 bpi. If you are using 1600 bpi, divide the numbers by two; if you are using 6250 bpi, double the numbers. Similarly, if you use a reel of tape with some other length, apply the appropriate ratio.

The sizes of the databases in a Statice File System are shown in Statice File System directory listings, in blocks. A block is 1152 bytes. So, for example, if you had a Statice File System containing four databases, two 1000 blocks long and two 2000 blocks long, the total amount of data is about 6 MB, which will easily fit on one tape of any kind.

Tertiary Volumes and Volume Sets

A *volume* means one physical tertiary storage medium, such as one reel or one cartridge of tape. (In future releases, a single write-once optical disk will also be referred to as a volume.)

A *volume set* is a set of one or more volumes. Volumes are grouped into sets because each volume is of a fixed size, whereas you can keep expanding the size of a volume set by adding more volumes.

Each complete dump of a database file system is put on its own volume set. The number of volumes in the volume set depends on the size of the database file system. If the Statice File System is not very large, a complete dump fits on a single volume, which constitutes a single volume set.

Each volume set has a *name*, and each volume within the volume set has a *sequence number*. The name is a character string. No two volume sets can have the same name. Names are compared ignoring case, so you must not have one volume named "Foo" and another named "foo". Volume set names may not use character styles or non-standard character sets. The first volume of a volume set should be numbered 1, and each sequential volume is assigned the next number.

When you're using the backup system and a tape is being mounted, you are prompted for a "mount specification". This consists of a volume set name, a sequence number, and a device specification. It means that you are mounting a particular volume on a particular device. To specify the device, you are prompted for a host name, a device type, and further parameters depending on the type of the device. For details of device specs: See the section "Choosing the Kind of Tertiary Storage to Use".

Labels on Volumes

When a volume is being used by the Statice File System backup system, some special identification information is written at the front of the volume, called the *label*. The label includes the volume set name and the sequence number, which uniquely identify the volume. The backup software uses the label to help assure that you have mounted the tape you intended to mount, in order to guard against mistakes.

When you first use a brand-new tape for Statice File System backup, you should write a label on it, by using the Initialize Backup Volume command in the Statice File System Operations program. Be careful to only use this command on fresh, unused tapes, because it will destroy any information already on the tape.

When the backup system is making a backup copy, and it is ready to write onto a new tape, it first reads the label of the tape, to make sure that this tape is the right one. For example, suppose you are making a backup copy to a volume set named FULL0013, and the backup system just finished writing volume number 2 of the volume. It prompts you with the message:

```
End of volume FULL0013/2.  Enter mount specifications for the next volume:
```

You enter a mount specifiction, in which you enter a volume set name and sequence number. The default is volume set FULL0013, sequence number 3. You also physically mount the correponding tape on the device that you specify in the mount specifiction. After you click on Done or press End, the backup system reads the volume label, to make sure that this volume is really volume number 3 of volume set FULL0013. If the label is present and contains what backup expects, the backup copy continues. Otherwise, you are given several options:

Accept the information on the volume
: (*You believe you entered the wrong thing, and you want to use what the tape says.*) This choice tells the backup system to use the volume set name and sequence number that were found in the label on the tape, even though they aren't what we originally expected. This sets the backup system's concept of "current volume", so the backup system will expect the next volume to follow this one.

Remount a different volume
: (*You believe you mounted the wrong tape, and wish to try mounting a different one.*) This choice lets you remove this volume from the drive and try another one. The backup system

prompts you with "Is the desired volume mounted?" When you answer "y", it proceeds to read the label of the new tape and check it.

Reenter different volume specifications

(*You believe you entered the wrong thing, and you want to try to enter it again.*) This choice makes the backup system return to the point where you were prompted for mount specifications for this tape; you are prompted again. Use this if the reason for the problem is that you didn't provide the right mount specification.

Overwrite the volume with the specified information

(*You believe that what the tape says is wrong, and you want to use what you entered.*) This choice tells the backup system to ignore what the label says, and write the tape using the mount specification that you provided. This overwrites the label and can change the volume set name and sequence number in the label.

Abort the current operation

(*You don't want to proceed further.*) This choice returns you to the Statice File System Operations program.

If the tape you mount is a fresh, unused tape, you'll get this list of choices, and you can select [Overwrite] to write a new label on the tape. So it is not actually necessary to use the Initialize Backup Volume command to write an initial label; [Overwrite] also does it for you. However, there's a drawback to relying on [Overwrite]. When the backup system tries to read the label of the blank tape, the tape drive attempts to read data from the tape, but cannot find any. Some drives react to this lack of data by reading further and further down the tape, hoping to find data eventually, which can take a long time. So if you don't run Initialize Backup Volume on new tapes, backup copying might be substantially slower.

This behavior is part of the tape drive and tape controller, and cannot be modified by software. Currently, all industry-standard tape drives sold by Symbolics exhibit this behavior, but the cartridge tape drives do not. Therefore, it is particularly time-saving to use Initialize Backup Volume on industry-standard tapes.

Volume Libraries

When you do Statice File System operations, Statice maintains a database called the volume library. This database is stored within the file system. The volume library stores the following information:

For every backup volume that holds information from this file system:

- Volume name and sequence number
- Completion date, which is the date and time this tape was last written
- Type, which is either "Industry Standard Tape" or "Cartridge Tape"

- The tape spec that was used when the tape was last written, which includes the host, and also unit and density for industry-standard tapes

For every backup run (that is, for every time that a dump as made):

- Completion date, which is the date and time this run was performed
- The set of volumes that were written.
- Whether the run is "valid", or whether something went wrong during the dump

For every file that has been dumped:

- Name of the file
- The set of backup-notes (see below)

For every copy of a file that exists on tape, a "backup note", consisting of:

- The file attributes (length, creation-date, author, comment)
- Which volume the copy resides on
- Which backup run this copy was part of

Using the Statice File System Operations Program

Before using the Statice File System Operations program, you must load it by entering:

```
Load System DBFS-Utilities
```

You enter the Statice File System Operations program by entering:

```
SELECT symbol-sh-D
```

The commands appear a menu in the top pane. You can click on them, or type the names of the commands. Typically, an AVV menu will be displayed, prompting you for several fields. When you finish entering the information and press `END`, the command is executed.

We summarize the commands here. For complete documentation on each command: See the section "Dictionary of Statice File System Operations Commands".

Backing Up and Restoring a File System

The most common operations are Complete Backup and Complete Restore. Selective Restore can also be used, to restore one or more databases from tape to the file system.

Complete Backup Command
: Copies a Statice File System (and all databases in it) to tape.

Complete Restore Command
: Copies a file system (and all databases in it) from a tape to a Statice File System.

Selective Restore Command
: Copies selected databases from a tape to a Statice File System.

Initialize Backup Volume Command
: Writes the volume number on the tape label itself.

Getting Information

Describe Statice File System Command
: Displays information about a file system.

Describe Backup Volume Command
: Displays information about a backup volume stored on a tape.

Show Backup History Command
: Displays information about all backup runs done on a file system.

Compare Backup Volume Set Command
: Compares a file system stored on tape with a file system stored on the local machine.

Enabling and Disabling a File System or All of Statice

Enable Statice File System Command
: Enables a file system: allows transactions to be started.

Disable Statice File System Command
: Disables a file system: aborts any transactions in progress and disallows any transactions to occur until the file system is enabled again.

Enable Statice Command
: Re-enables Statice activities.

Disable Statice Command
: Entirely disables all Statice activities.

File System Manipulation

Create Statice File System Command
: Creates a new Statice File System on the local host.

Delete Statice File System Command
: Expunges an entire Statice file system, and removes all traces of it, including every database in it; this is a very dangerous command.

Show All Statice File Systems Command
: Lists all the file system objects and the host that they reside on in a namespace.

Add Statice Partition Command
: Adds a new partition to an existing Statice file system.

Show Statice Partitions Command
: Lists the partitions of a Statice file system, and shows the amount of free space remaining in each partition.

Dictionary of Statice File System Operations Commands

This section documents the commands that are available only in the Statice File System Operations program.

The following commands are available from the Statice File System Operations program, but are described elsewhere because they are also available at top level.

- "Add Statice Partition Command"
- "Create Statice File System Command"
- "Delete Statice File System Command"
- "Show All Statice File Systems Command"
- "Show Statice Partitions Command"

For documentation on the other Statice commands that can be used at top level: See the section "Dictionary of Statice Commands".

Complete Backup Command

Complete Backup

Copies all databases in a Statice File System to tape. This command is available in the Statice File System Operations Program.

The AVV menu looks like this:

```
Enter complete backup parameters (volume name required):

File system to backup: DLW-UFS-3
Volume set name: abc
Volume sequence number: 1
Device: Industry-Standard-Tape  Cartridge-Tape
Volume host: ALDERAAN
Show Detailed Progress: Yes  No
```

The file system to backup must be stored on the local host. We discuss volume set names and sequence numbers elsewhere: See the section "Labels on Volumes".

The volume host is the host that will write the data to tape; it need not be the local host. If that host does not have TAPE service in its namespace object, you will get this error:

```
Error:  Host does not support TAPE service.
```

You can use one of the proceed options to try TAPE service on various mediums such as TCP or CHAOS. You can also edit the host's namespace object to add that service entry, so the error won't happen again.

Symbolics recommends RTAPE via TCP rather than RTAPE via ChAOS (TCP is more reliable). Add the service `TAPE TCP RTAPE` to the host's namespace object.

If you are writing to a blank tape, you will get an error stating that the volume set name does not match that of the tape itself. This is to be expected (because the tape doesn't have a volume set name at all yet). One way to prevent this error is to use the Initialize Backup Volume command before doing Complete Backup, to write the volume set name on the tape. In any case, one of the choices is to Overwrite the tape, which is the correct choice for a blank tape.

Compare Backup Volume Set Command

Compare Backup Volume Set

Compares a file system stored on tape with a file system stored on the local machine. This command is available in the Statice File System Operations Program.

The AVV menu looks like this:

```
Enter specifications for compare:

File system to compare: DLW-UFS-3
Volume set name: abc
Volume sequence number: 1
Device: Industry-Standard-Tape  Cartridge-Tape
Volume host: ALDERAAN
Show Detailed Progress: Yes  No
```

If anyone has made changes to a database in the file system since the backup was done, Statice will report how many pages are different. The output looks like this:

```
File SCRC|DLW-UFS-3:>Volume-Library>%volume-library has 21 pages
different from the tertiary image.
These differences could be caused by updates which occured
between the time the backup volume finished writing and the time
the comparison finished.
```

Complete Restore Command

Complete Restore

Copies all databases from a tape to a Statice File System. This command is available in the Statice File System Operations Program.

The AVV menu looks like this:

```
Enter specifications for restore:
```

```
File system to restore: DLW-UFS-3
Volume set name: abc
Volume sequence number: 1
Device: Industry-Standard-Tape  Cartridge-Tape
Volume host: ALDERAAN
Show Detailed Progress: Yes  No
```

The choices are the same as for the Complete Backup command: See the section "Complete Backup Command".

The first thing that a Complete Restore does is delete all databases in the file system. It then copies the databases from tape to the file system. Note that if you are writing to a file system that already exists, you will get this message:

```
Before a file system can be restored, the existing files must be
destroyed.  All the files in file system SCRC|DLW-UFS-3 are
about to be destroyed.  Are you sure you want to continue? (Yes
or No)
```

If you want to keep the existing file system, choose No. If you want to restore the file system from tape, choose Yes.

Describe Backup Volume Command

Describe Backup Volume

Displays information about a backup volume stored on a tape. This command is available in the Statice File System Operations Program.

This command gets its information from the tape itself.

The AVV menu requests the type of tape (whether cartridge or industry standard), and the volume host (the host where the tape is mounted).

Describe Statice File System Command

Describe Statice File System *file-system-name*

Displays information about a file system. This command is available in the Statice File System Operations Program.

This command gets information from the file-system namespace object. If the file system is local, it also displays information about how much data is stored in each partition.

Disable Statice Command

Disable Statice

Entirely disables all Statice activities:

- Every currently-running transaction is aborted, signalling the error **dbfs::system-shutdown-transaction-abort**, whose error message is "Transaction aborted due to a system shutdown."
- All server processes executing on the local host are killed.
- All local file systems are shut down. See the section "Disable Statice File System Command".
- The local host is marked as disabled, so that any new file systems created on the host are automatically disabled.
- All network connections associated with Statice services are killed.

Any attempts to start a transaction while Statice is disabled signal an error. Any attempts by a remote host to connect to this host are rejected. This command is available in the Statice File System Operations Program.

Disable Statice File System Command

Disable Statice File System *file-system-name*

Disables a file system: aborts any transaction in progress that owns a lock on any page of any file in the file system. Disallows any transactions to occur until the file system is enabled again. If any transaction attempts to use a database in the file system, the error **dbfs::file-system-disabled** is signalled. This command is useful if you want to perform administrative activities on a database, and want to make sure nobody else accesses the database. Note that it is not necessary to disable a Statice file system to do a backup dump.

file-system-name must be the name of a Statice file system on the local host. This command is available in the Statice File System Operations Program.

Enable Statice Command

Enable Statice

Re-enables Statice activities. Undoes the effect of Disable Statice, returning Statice to normal. This command is available in the Statice File System Operations Program.

Enable Statice File System Command

Enable Statice File System *file-system-name*

Enables a file system: allows transactions to be started. Undoes the effects of Disable Statice File System, returning the file system to normal. This command is available in the Statice File System Operations Program.

Initialize Backup Volume Command

Initialize Backup Volume

Writes the volume number on the tape label itself. This command is available in the Statice File System Operations Program.

This command is not a necessary step, because Complete Backup will also write the information on the tape, but it prevents the mismatch that can occur when writing a blank tape: the volume number you specify won't match that of the tape (because a blank tape won't have any volume number on it).

Selective Restore Command

Selective Restore

Copies selected databases from a tape to a Statice File System. This command is available in the Statice File System Operations Program.

The AVV menu looks like this:

```
Enter specifications for selective restore:

File system: DLW-UFS-3
Disable file system: Yes  No
Paths to restore: >test2, >test7, and >test8
Repatriate action: Yes  No  Query
Name conflict resolution action: Leave
                                 Rename Existing File
                                 Replace Existing File
                                 Load Into Unique File
                                 Query
Volume selection mode: Automatic  Manual
Device: Industry-Standard-Tape  Cartridge-Tape
Volume host: ALDERAAN
Show Detailed Progress: Yes  No
```

If you choose Yes for "Disable file system", Statice does a Disable Statice File System before doing the Selective Restore, and an Enable Statice File System afterwards. This option is useful if you are performing some kind of delicate operation on the file system, such as recovering from a problem, and you want to make sure that no other users make any changes in the file system while the Selective Restore is in progress. In the general case, it is not necessary to do this.

The "Paths to restore" are pathnames of databases within the file system. They should start with the greater-than sign, >. They are separated by commas. These pathnames can contain wildcards.

Once you press END, Statice searches the volume libary to figure out which volume the file is on. You will then be prompted to mount that volume:

```
Is volume abc/1 mounted for restoring? (Y or N)
```

When Selective Restore is searching the volume library to figure out which volume to retrieve a file from, there might be more than one volume that has this file on it. In such a case, it chooses the volume with the most recent completion date (the date and time at which the dump was completed); that is, it uses the most recent backed-up copy.

The repatriation action should never be needed by applications programmers, so you should use the default (no) for this choice.

Show Backup History Command

Show Backup History *file-system-name*

Displays information about all backup runs done on a file system. This command is available in the Statice File System Operations Program.

This command gets its information from the volume library:

Volume Libraries

When you do Statice File System operations, Statice maintains a database called the volume library. This database is stored within the file system. The volume library stores the following information:

For every backup volume that holds information from this file system:

- Volume name and sequence number
- Completion date, which is the date and time this tape was last written
- Type, which is either "Industry Standard Tape" or "Cartridge Tape"
- The tape spec that was used when the tape was last written, which includes the host, and also unit and density for industry-standard tapes

For every backup run (that is, for every time that a dump as made):

- Completion date, which is the date and time this run was performed
- The set of volumes that were written.
- Whether the run is "valid", or whether something went wrong during the dump

For every file that has been dumped:

- Name of the file

- The set of backup-notes (see below)

For every copy of a file that exists on tape, a "backup note", consisting of:

- The file attributes (length, creation-date, author, comment)
- Which volume the copy resides on
- Which backup run this copy was part of

This command first iterates over all the backup runs, in order of completion date (most recent first). For each run, it tells you whether the run is valid, and what the completion date is. Then it iterates over all the volumes in that backup run, sorted by completion date. For each volume, it prints out the name/sequence-number, the type, the completion date, the host, the unit (if any), and the density (if any).

High-level Dumper/Loader of Statice Databases

The Dump Database and Load Database commands invoke a "high-level" database dump/load tool. High level means that it dumps and restores the data in a "source" format, rather than in a binary page format as does the dump/restore tool available from the Statice Operations Menu. The high-level dump/load tool can used for certain types of database reorganization operations.

Dump Database always dumps the databse in a transaction-consistent state, because it uses one long transaction to do its job.

The high-level dumper/loader is useful for several purposes. It enables you to:

- Move a database from one place to another, over a channel that can handle only ordinary text.
- Store the contents of a database on some kind of storage medium (e.g. a particular tape format) that can handle only ordinary text.
- Edit the text file to reorganize the database.

Limitations of the High-level Dumper/Loader

- It does not dump to tape, so the size of the dump is limited to the amount of available file server disk space. Further, since the format is "source" level, it may actually take more disk space to dump a database using the Dump Database command than it does to store it.
- You shouldn't do dumping while other users are operating on the database, since it dumps everything in one large transaction (which will lock them out, or else cause the dump/restore facility to abort a transaction). Loading data back into a database can be rather slow since it does many small transactions (to avoid growing the log).

- Because of LMFS file size limitations, it may not be appropriate to dump a database to a LMFS file. This size limitation is approximately 15MB. Instead, you may have to dump to a UNIX or VMS machine with enough disk space.

Clustering is Maintained

Clustering is maintained across dumps. The actual entities may be grouped differently within the clusters, but they will all be in the same cluster that they were before.

Format of the Dump File

The dump file consists of lists which contain information about the contents of the database. At the beginning of the file is information about the real schema, and following that is information about the actual contents of the database. The format of each list is that the first element of it is a keyword symbol specifying what the list is about and the rest of the list is information specific to that type of list.

Schema information is contained in lists which begin with the :DOMAIN, :RELATION, :COMMIT-DOMAINS, and :INDEX keywords. For the most part, users should not modify these lines unless it is obvious what they mean. For example, consider the following definition, which is taken from the file SYS:STATICE;EXAMPLES;BOOKS.LISP:

```
(define-entity-type account ()
  ((name string :unique t :cached t :inverse account-named)
   (balance single-float :cached t)
   (type (member checking owner) :cached t)))
```

The corresponding information in the dump file looks like this:

```
(:RELATION "ACCOUNT" 1 NIL (("%$OF" "ACCOUNT" T T NIL NIL NIL) ("TYPE"
 (CL:MEMBER BOOKS:CHECKING BOOKS:OWNER) NIL NIL NIL NIL NIL) ("BALANCE"
 CL:SINGLE-FLOAT NIL NIL NIL NIL NIL) ("NAME" STRING T NIL NIL NIL NIL)))
```

This information appears on one line in the dump file. If we wanted to change the balance attribute's data type from single-float to double-float, then we'd edit the "CL:SINGLE-FLOAT" piece of text above. Of course if you change the data type of an element like this, you'd also have to change all the data in the file, too.

After the schema information, the actual data in the file is dumped to the file using :ENTITY and :RELATION-DATA entries. The format of an :ENTITY entry is:

```
(:ENTITY ("ENTRY" 14352 3388287 7463818 3147232 NIL))
```

This is a list of an entity's type name (ENTRY), its internal record ID (14352), its unique ID (three 32-bit fixnums), and its cluster ID. For the most part, these should be of little interest to the users.

Users are more likely to be interested in the :RELATION-DATA entries. An account entity might look like this:

```
(:RELATION-DATA "ACCOUNT" NIL (17441 BOOKS:OWNER 200.53 "Lane"))
```

To change the value of the balance for the "Lane" account, you'd change the value 200.53 to another value. You can find out the order of the attributes of this by looking at the real-schema data at the beginning of the file. The attribute order information will be embodied in the :RELATION entry. For example, the :RELATION line above shows that the order of the attributes in the above :RELATION-DATA entry are %$OF (an internal attribute), the type attribute, the balance attribute, and the name attribute.

For more detailed information, you can read the comments in the code in the file SYS:STATICE;UTILITIES;MODEL-DUMPER.LISP.

Dictionary of Statice Commands

This section documents the commands that are available in the command processor. Statice also offers a set of commands that are available only in the Statice file System Operations menu. For documentation on those commands: See the section "Dictionary of Statice File System Operations Commands".

Add ASYNCH DBFS PAGE Service Command

Add ASYNCH DBFS PAGE Service *host-name keywords*

Updates a host's namespace object to contain the ASYNCH-DBFS-PAGE service.

keywords :TCP Not Present

:TCP Not Present
{Yes No} If yes, no service entry is added for the TCP medium

Statice uses the ASYNCH-DBFS-PAGE service for communicating various signals and commands back to each of the client hosts, and hence should be present on all Statice clients. It need not be present on Statice servers however, unless they are clients to some other server.

This command adds the service-medium-protocol triplet for both the TCP and CHAOS mediums to the namespace object for the host. You should only need to perform this command once, when the file system is installed on a server. If the host does not support TCP, supply Yes to the :TCP Not Present keyword option.

Add DBFS PAGE Service Command

Add DBFS PAGE Service*host-name keywords*

Updates a host's namespace object to contain the DBFS-PAGE service.

keywords :TCP Not Present

:TCP Not Present
{Yes No} If yes, no service entry is added for the TCP medium.

Statice uses the DBFS-PAGE service uses for communicating database pages and requests over the network, and hence should be present on all Statice File System server hosts. It need not be present on client hosts, however.

This command adds the service-medium-protocol triplet for both the TCP and CHAOS mediums to the namespace object for the host. You should only need to perform this command once, when the file system is installed on a server. If the host does not support TCP, supply Yes to the :TCP Not Present keyword option.

Add Statice Partition Command

Add Statice Partition *file-system-name partition-pathname size*

Enables you to add partitions dynamically to a Statice file system (e.g. when it is running out of space).

file-system-name Name of a Statice file system that is stored on the local host.

partition-pathname Pathname of a partition; this pathname must name a FEPFS file on the local host (although the file need not exist).

size The size of the new partition, in blocks.

This command creates the new partition, allocates the space from the size given, and makes the new partition available to Statice for allocating.

This command may be given when a Statice server process receives a file system full error. For example, if a server process signals the following error, you can add a new partition and resume the operation:

```
Error: The File System "Squash" is full.

(FLAVOR:METHOD UFS::FIND-FREE-BLOCKS UFS:UFS-FILE-SYSTEM-MIXIN)
 proceed options...

 :Add Statice Partition (a file-system) SQUASH
 (the pathname of a file) FEP2:>Statice>SQUASH-part2.file.newest
 (Size in blocks [default 1000]) 1000

Updating file-system object SCRC|SQUASH in namespace... Done
```

After adding the partition, select the proceed option that resumes the operation.

Copy Statice Database Command

Copy Statice Database *from-database to-database keywords*

Copies all the pages of one database to the other (possibly new database) inside a transaction.

from-database Pathname of the database to copy.

to-database Pathname of the destination, where the database should be copied.

keywords :Copy Properties, :Create Directories, :Query

:Copy Properties {any combination of: Author, Comments, Creation Date} This indicates which properties should be copied to the new database(s). The default is Author and Creation Date.

:Create Directories {Yes, Error, Query} Yes means that directories that do not exist should be created silently, Query will ask, and Error will cause an error if they do not exist.

:Query {Yes, No, Ask} Whether to ask before copying each file.

Create Statice File System Command

Create Statice File System *file-system-name keywords*

Creates a new Statice File System on the local host. You cannot use this command to create a file system on a remote host.

file-system-name A symbol naming the new file system.

keywords :Locally

:Locally {Yes, No} Whether to update only local namespace information (Yes), or to update the namespace database server as well (No). The default is No. See the section "The Locally Namespace Editor Command".

The command displays an AVV menu in which you specify the names of various parameters. Above the menu, you will see a list of all the disk drives on the local host and the amount of free space available on each of them. The AVV menu asks for the following items:

Directory Partition: This entry specifies the FEPFS file in which the internal file system directory resides. Its size is determined by the number of directory entries which you specify in the Maximum Directory Entries field. The default file name for the directory partition is FEP*n*:>Statice>*fs-name*-part*m*.UFD. *n* is the highest mounted disk on the system, *fs-name* is the file-system name

specified for the command, and *m* is the number of the partition.

Maximum Directory Entries:
: This entry specifies the maximum number of databases which may reside in the file system at any time. Note that Statice always takes two of these entries for itself—one for the log file, and one for the Directory database. These entries are reusable, so if a database is deleted, using the Delete File command (in conjunction with a database pathname, not a FEPFS pathname), that entry in the file system directory is reusable for another database. On the Symbolics 36xx, there are 71 directory entries in each FEPFS block. The directory is organized as a hash file, so it's desirable to make the directory large enough that it's not densely filled.

Partition:
: These entries specify the partitions to be used for the file system. There may be as many partitions as you want, and they can live on any of the disks. In general, there should be as few partitions as possible in order to avoid disk fragmentation. The default pathname for a partition is FEP*m*:>Statice>*fs-name*-part*n*.file, where *m* is the highest mounted unit number, *fs-name* is the name of the file system, and *n* is the partition number in the ordering of all the partitions entered in the AVV menu.

Blocks:
: This entry specifies the number of blocks to allocate for the partition. When you enter a value for this field, the values in the available disk space headings will change accordingly to take into account how much of the free space you have allocated. You may click on None in this field to remove the partition the menu (and hence not include it as part of the file system when it is created).

When all the parameters have been entered, pressing `END` will cause the file system to be created. First, the file system object will be created in the namespace database (permanently, unless :Locally Yes was specified). Note that the messages printed by the command do not indicate whether the namespace was updated locally or globally. Second, all of the partitions are created in the FEPFS, and their :DONT-DELETE properties are set. You don't need to create any of the partitions yourself—this is done automatically for you, including the proper allocation of space. Third, the log file in the file system is initialized. Finally, the directory database is created.

Here's a sample run:

```
Command: Create Statice File System SQUASH
FEP0: 21464 Available (Originally: 21464 free, 88696/110160 used (81%))
FEP1:   137 Available (Originally: 137 free, 146743/146880 used (100%))
FEP2: 70727 Available (Originally: 71742 free, 38418/110160 used (35%))
```

```
Directory Partition: FEP2:>Statice>SQUASH.UFD
Maximum Directory Entries: 1000
Initial Log Size in Blocks: 500
Partition: FEP2:>Statice>part0.file.newest
  Blocks (None to remove): None  1000
Partition: FEP2:>Statice>part1.file.newest
  Blocks (None to remove): None  an integer

Creating file-system object SCRC|SQUASH in namespace... Done.
Initializing local UFS with associated directory structure... Done.
Creating local DBFS with associated directory structure... Done.
Initializing DBFS Directory database... Done.
```

Delete Statice File System Command

Delete Statice File System *file-system-name keyword*

Expunges an entire Statice file system, and removes all traces of it, including every database in it; this is a very dangerous command.

file-system-name A symbol naming a file system that is resident on the local host.

keywords :Locally

:Locally {Yes, No} Whether to update only local namespace information (Yes), or to update the namespace database server as well (No). The default is No. See the section "The Locally Namespace Editor Command".

Because this command permanently removes the Statice File System, and all databases in it, it is a dangerous command and it asks for confirmation. If you answer Yes, the file system and all the databases in it are destroyed by removing the file system partitions from the FEP directory in which they were placed by the Create Statice File System command. The command destroys the file system partitions, even though they may have the :DONT-DELETE flag set for them in the FEPFS (the Create Statice File System command sets the :DONT-DELETE property for each of the partitions in a file system).

If you have done a complete backup dump, you can restore the contents of a deleted file system by using the Complete Restore command of the Statice File System Operations activity. If you have not done a complete backup, the data cannot be restored.

Dump Database Command

Dump Database *database-pathname destination-pathname*

Writes all the information in the database into a text file.

database-pathname A pathname indicating the location of a Statice database.

destination-pathname
A pathname of a file on any file system; this need not be stored on a Symbolics machine.

This command is useful for several purposes. It enables you to:

- Move a database from one place to another, over a channel that can handle only ordinary text.
- Store the contents of a database on some kind of storage medium (e.g. a particular tape format) that can handle only ordinary text.
- Edit the text file to reorganize the database.

We discuss the details of the Dump Database and Load Database commands elsewhere: See the section "High-level Dumper/Loader of Statice Databases".

Load Database Command

Load Database *database-pathname destination-pathname keywords*

Takes a text file produced by Dump Database, and makes a new database containing the same information.

database-pathname A pathname indicating the location of a Statice database.

destination-pathname
A pathname of a file on any file system; this need not be stored on a Symbolics machine.

keywords :If Exists

:If Exists {Error, Create} Specifies the action to be taken if the database specified by the *database-pathname* already exists. Error signals an error, and Create causes the old database to be erased and replaced by the database being loaded.

Unless you have edited the text file, Load Database makes an exact copy of the original database that was dumped, including keeping the unique ids the same.

We discuss the details of the Dump Database and Load Database commands elsewhere: See the section "High-level Dumper/Loader of Statice Databases".

Set Database Schema Name Command

Set Database Schema Name *pathname new-schema-name*

Informs the database that its schema name is now the given *new-schema-name*.

pathname A pathname indicating the location of a Statice database.

new-schema-name A symbol.

If you move a Statice program from one package to another, and the database already exists, it is necessary to use this command to update the database to inform it of the new schema name.

See the section "Warning About Changing the Package of a Statice Program".

Show All Statice File Systems Command

Show All Statice File Systems *namespace*

Lists all the file system objects in the namespace, and the host on which each one resides.

namespace A symbol that specifies a namespace in which to search. By default, all namespaces in the namespace search path are searched

Show Database Schema Command

Show Database Schema *pathname*

Prints the definition of the schema of the database specified by pathname. This command is useful if you see a database in a Statice file system and don't know what it is. It's also useful for seeing what indexes currently exist in a database.

Not all of the information from the template schema is stored in the database itself, so when Show Database Schema reconstructs the schema definition from the database, not all of the original information is recovered. Specifically:

The following attribute options are reconstructed: **:unique**, **:index**, **:index-average-size**, **:inverse-index**, **:inverse-index-average-size**, **:inverse-index-exact**, **:inverse-cached**, **:area**, **:attribute-set**, and **:no-nulls**.

The following attribute options are not reconstructed: **:cached**, **:initform**, **:inverse**, **:inverse-exact**, **:cluster**, **:accessor**, **:reader**, **:writer**, and **:read-only**.

The following entity-type options are reconstructed: **:area**, **:type-set**, **:multiple-index**, and **:multiple-index-exact**.

The following entity-type options are not reconstructed: **:conc-name**, **:constructor**, **:default-init-plist**, **:documentation**, **:init-keywords**, **:instance-variables**, and **:own-cluster**.

See the section "Examining the Schema of a Statice Database".

Show Statice Partitions Command

Show Statice Partitions *file-system-name*

Shows the amount of free space remaining in a file system's partition(s).

file-system-name Name of a Statice file system which resides on the local host.

This command may be done only for a file system stored on the local host.

For example:

```
Show Statice Partitions (a file-system) SQUASH

Partition                                   Free Space
FEP1:>squash>squash.file.newest             0/1056
FEP0:>squash>squash.file.newest             0/3000
FEP2:>Statice>SQUASH-part2.file.newest      54/1000
```

Update Database Schema Command

Update Database Schema *database-pathname*

Used when you have modified a schema; this command compares the template schema to the real schema in the database, and updates the real schema to match the template schema.

database-pathname A pathname indicating the location of a Statice database.

We discuss this subject in detail elsewhere: See the section "Modifying a Statice Schema".