Sun StorEdge™ SAM-FS 存储和存档管理指南

版本 4.1

Sun Microsystems, Inc.
www.sun.com

文件号码 817-7390-10
2004 年 6 月，修订版 A
请将有关本文档的意见或建议提交至：http://www.sun.com/hwdocs/feedback

目录

序言

本手册 *《Sun StorEdge™ SAM-FS 存储和存档管理指南》*，介绍 Sun StorEdge SAM-FS 4.1 版本所支持的存储和存档管理软件。Sun StorEdge SAM-FS 软件可以自动将文件从在线磁盘复制到存档介质中。存档介质可由在线磁盘或可移动介质卡盒组成。

Sun StorEdge SAM-FS 4.1 版在以下 Sun Solaris™ 操作系统 (OS) 平台上受支持：

- Solaris 8, update 5, 7/01
- Solaris 9, update 3, 4/03

本手册旨在针对负责配置和维护 Sun StorEdge SAM-FS 软件的系统管理员。并假定系统管理员非常熟悉 Solaris OS 过程，包括创建帐户、执行系统备份和其它基本的 Solaris 系统管理任务。

注意 – 如果要使用 Sun StorEdge SAM-FS 软件中提供的存储和存档管理程序来运行 Sun StorEdge QFS 文件系统，那么可以购买 Sun StorEdge QFS 和 Sun StorEdge SAM-FS 软件的许可证。这类系统称为 *Sun SAM-QFS*。

除非在必须明确以避免混淆的情况下，本手册并未列出 Sun SAM-QFS 配置。在本手册中谈到存储及存档管理时，您可以假定对 Sun StorEdge SAM-FS 的参考同样适用于各种 Sun SAM-QFS 配置。同样，当谈到文件系统设计和性能时，您可以假定对 Sun StorEdge QFS 的参考同样适用于各种 Sun SAM-QFS 配置。

本书的内容编排

本书包括以下章节：

- 第一章，产品概述。
- 第二章，介绍基本操作。本章所述的信息适用于大多数自动化库和手动载入的设备。
- 第三章，介绍如何依据各类库的专用操作说明来管理相应库中的卡盒。这一章还介绍了这些库及其相应的基本操作步骤。
- 第四章，说明存档过程。
- 第五章，说明释放过程。
- 第六章，说明登台过程。
- 第七章，说明回收过程。
- 第八章，说明专门针对 Sun StorEdge SAM-FS 环境的升级过程。
- 第九章，说明有关 Sun StorEdge SAM-FS 操作的高级主题。

词汇表，定义了本手册及其它 Sun StorEdge QFS 和 Sun StorEdge SAM-FS 文档中所用到的术语。

使用 UNIX 命令

本文档没有介绍基本的 UNIX® 命令和操作过程，如关闭系统、启动系统和配置设备等。有关此类信息的说明，请参阅以下一个或多个文档：

- 系统附带的软件文档
- Solaris™ 操作系统的有关文档，其 URL 如下：

  `http://docs.sun.com`

Shell 提示符

表 P-1 列出了本手册中使用的 shell 提示符。

表 P-1 Shell 提示符

Shell	提示符
C shell	*计算机名* %
C shell 超级用户	*计算机名* #
Bourne shell 和 Korn shell	$
Bourne shell 和 Korn shell 超级用户	#

印刷惯例

表 P-2 列出了本手册采用的印刷惯例。

表 P-2 印刷惯例

字体或符号	含义	示例
AaBbCc123	命令、文件和目录的名称；计算机屏幕输出信息。	编辑 .login 文件。 使用 ls -a 列出所有文件。 % You have mail.
AaBbCc123	键入的内容（相对于计算机屏幕输出信息）。	% **su** Password:
AaBbCc123	书名；新词或术语；需要强调的字词；以及命令行中需用实名或实际值替换的变量。	参阅 *“用户指南”* 的第六章。 这些称为*类*选项。 您*必须*是超级用户 (root) 才能执行此操作。 若要删除文件，请键入 rm *文件名*。

表 P-2　印刷惯例 *(接上页)*

字体或符号	含义	示例
[ ]	在命令语句中，方括号内的参数表示可选参数。	`scmadm [-d sec] [-r n[:n][,n]...][-z]`
{ *arg* | *arg* }	在命令语句中，大括号和竖线表示必须指定其中一个参数。	`sndradm -b { phost | shost }`
\	命令行末尾的反斜杠 (\) 表示此命令续接下一行。	`atm90 /dev/md/rdsk/d5 \ /dev/md/rdsk/d1`

相关文档

本手册是介绍 Sun StorEdge QFS 和 Sun StorEdge SAM-FS 软件产品操作文档系列的一部分。表 P-3 列出了这些 4.1 版产品的完整文档系列。

表 P-3　相关文档

书名	文件号码
《Sun SAM-Remote 管理员指南》	816-8737-11
《Sun QFS、 Sun SAM-FS 和 Sun SAM-QFS 故障恢复指南》	816-7680-10
《Sun StorEdge QFS 和 Sun StorEdge SAM-FS 文件系统管理指南》	817-7385-10
《Sun StorEdge QFS 和 Sun StorEdge SAM-FS 软件安装和配置指南》	817-7395-10
《Sun StorEdge SAM-FS 存储和存档管理指南》	817-7390-10
《Sun StorEdge QFS 和 Sun StorEdge SAM-FS 发行说明》	817-7400-10

访问 Sun 在线文档

Sun StorEdge QFS 和 Sun StorEdge SAM-FS 软件销售套件中包括了这些产品文档的 PDF 文件。用户可从以下位置查看这些 PDF 文件：

■ Sun Network Storage 文档网站。

此网站包含许多存储软件产品的文档。

a. 若要访问该网站，请输入以下 URL：

www.sun.com/products-n-solutions/hardware/docs/Software/Storage_Software

屏幕上会出现 Storage Software （存储设备软件）网页。

b. 从以下列表中单击适当的链接：

- Sun StorEdge QFS 软件
- Sun StorEdge SAM-FS 软件

- docs.sun.com。

此网站包含 Solaris 和其它多个 Sun 软件产品的文档。

a. 若要访问该网站，请输入以下 URL：

docs.sun.com

屏幕上会出现 docs.sun.com 网页。

b. 通过在搜索框中搜索以下项目之一来查找适用的产品文档：

- Sun StorEdge QFS
- Sun StorEdge SAM-FS

第三方网站

Sun 对本文档中所提及的第三方网站的可用性不承担责任。 Sun 也不对这些网站或资源上或由此获得的任何内容、广告、产品或其它资料，做出任何担保或承担任何责任。 Sun 不会对因使用通过这些网站或资源所获得的任何内容、商品或服务所带来的直接或间接的、实际的或声称的任何损害承担任何责任。

联系 Sun 技术支持

如果您遇到在本文档中无法解决的技术问题，请访问以下网址：

http://www.sun.com/service/contacting

使用许可

有关获取 Sun StorEdge QFS 和 Sun StorEdge SAM-FS 软件许可证的信息，请与 Sun 销售代表或授权的服务供应商 (ASP) 联系。

诊断程序

Sun StorEdge QFS 和 Sun StorEdge SAM-FS 软件中包含了 `samexplorer`(1M) 脚本。此诊断脚本对您和 Sun 客户支持人员非常有用。它不仅可以生成服务器配置的诊断报告，而且还可收集日志信息。安装软件之后，您可以访问 `samexplorer`(1M) 手册页来了解此脚本的详细信息。

安装帮助

要获得安装和配置服务，请拨打 1-800-USA4SUN 联系 Sun 企业服务部门，或联系当地的企业服务销售代表。

Sun 欢迎您提出宝贵意见

Sun 致力于提高文档资料的质量，并十分乐意收到您的意见和建议。可以将您的意见和建议提交至以下网址：

`http://www.sun.com/hwdocs/feedback`

请在您的反馈中包含本文档的书名和文件号码 （*《Sun StorEdge SAM-FS 存储和存档管理指南》*，文件号码 817-7390-10）。

第一章

概述

Sun StorEdge SAM-FS 环境提供了具有存储、存档管理以及检索功能的可配置文件系统。 Sun StorEdge SAM-FS 软件对文件进行存档的方法是，将文件从在线磁盘高速缓存复制到存档介质。存档介质可以是另一文件系统中的磁盘位片，也可以是自动或手动载入的存储设备中的可移动磁带或磁光盘卡盒。另外， Sun StorEdge SAM-FS 软件可自动将在线磁盘空间大小维持在站点指定的使用阈值上。它可以释放与已存档的文件数据相关联的磁盘空间，并在需要时将文件恢复到在线磁盘。

本章从技术层面对 Sun StorEdge SAM-FS 组件进行概述。它包括下列主题：

性能

Sun StorEdge SAM-FS 环境包括文件系统和存储及存档管理软件。Sun SAM-QFS 环境包括 Sun StorEdge QFS 文件系统。这两种文件系统均是位于服务器磁盘高速缓存中的高性能 UNIX 文件系统。这些文件系统之间的主要区别在于 Sun SAM-QFS 文件系统可以提供分布式共享文件系统等更高级的性能。有关文件系统自身的更详细信息，请参阅 *《Sun StorEdge QFS 和 Sun StorEdge SAM-FS 文件系统管理指南》*。

Sun StorEdge SAM-FS 环境中还包含以下组件：

- 存档程序，自动将在线磁盘高速缓存中的文件复制到存档介质中。存档介质可由在线磁盘文件或可移动介质卡盒组成。
- 释放程序，通过释放那些符合条件的已存档文件占用的磁盘块，自动将文件系统的在线磁盘高速缓存大小维持在站点指定的使用百分比阈值上。

- 登台程序，将文件数据恢复到磁盘高速缓存中。当用户或进程请求那些已从磁盘高速缓存中释放的文件数据时，登台程序会自动将文件数据复制回在线磁盘高速缓存中。
- 回收程序，清除包含已过期存档副本的存档卷，以使这些卷可重新使用。

以下几节将逐一简要地介绍这些功能。而后面几章将对它们进行更为详细的介绍。

存档

默认情况下，存档程序自动地为 Sun StorEdge SAM-FS 文件系统中的所有文件创建存档副本，并将存档副本写入存档介质。您可以配置存档程序，使其在不同存档介质上最多可创建 4 份存档副本。如果文件被*分成几段*，则每段都会被视为一个文件，且单独进行存档。当基于磁盘的文件符合站点定义的一套选择标准后，存档进程将会启动。

有关存档程序的详细信息，请参阅第 49 页的 “存档”。有关分段文件的详细信息，请参阅第 186 页的 “分段文件”。

释放

*释放*是指释放由已存档文件的数据占用的主 （磁盘）存储空间的过程。它使用两个阈值 （用磁盘总空间的百分比表示）来管理在线磁盘高速缓存的可用空间。这两个阈值分别是上限和下限。当在线磁盘占用空间超出上限时，系统会自动开始释放那些符合条件的已存档文件占用的磁盘空间。当在线磁盘占用空间到达下限时，系统会自动停止释放由已存档文件的数据占用的磁盘空间。选择释放文件的依据是文件的大小和文件在在线磁盘中存档的时间。另外，文件的第一部分可保留在磁盘上，以加快访问速度和掩饰登台过程的延迟。如果某个文件被分段存档，则可以单独释放该文件的各个部分。有关释放程序的详细信息，请参阅第 125 页的 “释放”。

登台

如果某个文件的数据块已被释放，则在系统访问该文件时，登台程序会自动将该文件或文件段的数据重新登台到在线磁盘高速缓存中。一旦开始登台操作，读取操作也将随即开始，从而使应用程序可以立即使用文件，而不必等到整个文件完全登台到磁盘高速缓存中。

Sun StorEdge SAM-FS 软件可自动处理登台请求错误。如果返回登台错误，系统会尝试查找文件的下一个可用存档副本。系统可自动处理的登台错误包括介质错误、介质不可用、自动化库不可用以及其它错误。有关登台的详细信息，请参阅第 143 页的 “登台”。

回收

一旦用户修改了文件，存档介质上与这些文件的旧版本相关联的存档副本将被视为*过期*。这些副本不再有用，因此可以从系统中清除。回收程序可以识别那些其中绝大部分是过期存档副本的存档卷，并将这些卷中的非过期副本移动到其它卷中进行保存。

如果可移动介质卷中仅包含过期副本，那么可以执行以下某一操作：

- 重新标记该卷，以便可立即重用。
- 将该卷的内容导出到离站存储设备中，作为文件变更的历史记录。可使用标准的 UNIX 实用程序从过期的存档副本中恢复文件的以前版本。

由于回收过程与最终用户的数据文件有关，因此它对最终用户是完全透明的。有关回收的详细信息，请参阅第 159 页的 “回收”。

存储设备

Sun StorEdge SAM-FS 环境可支持的磁带存储设备和磁光设备种类非常多。 Sun StorEdge SAM-FS 所支持的自动化库可根据其与环境连接的方式，分为以下几组：

- 直接连接。直接连接的自动化库使用小型计算机系统接口 (SCSI) 直接连接到主机系统。这种连接可以是直接连接，也可以是光纤信道连接。例如， Sun StorEdge 库使用直接连接方式。Sun StorEdge SAM-FS 系统直接使用自动化库的 SCSI 标准控制这些库。
- 网络连接。 Sun StorEdge SAM-FS 软件可配置为库的主机系统的客户机。网络连接自动化库包括 StorageTek 库、ADIC/Grau 库、IBM 库和 Sony 库等。这些库使用供应商提供的软件包。这时， Sun StorEdge SAM-FS 软件使用自动化库的专用后台程序，与供应商软件进行接口。

表 1-1 列出了不同自动化库专用的后台程序。

表 1-1 自动化库后台程序

后台程序	说明
sam-robotsd	监视传输器控制后台程序的执行情况。 sam-robotsd 后台程序由 sam-amld 后台程序自动启动。
sam-genericd	控制直接连接的库和介质更换器。还通过 DAS 接口控制 ADIC 库。
sam-stkd	通过 ACSAPI 接口控制 StorageTek 介质更换器。
sam-ibm3494d	通过 lmcpd 接口控制 IBM 3494 磁带库。
sam-sonyd	通过 DZC-8000S 接口控制 Sony 网络连接自动化库。

有关受支持的存储设备列表，请联系 Sun Microsystems 销售代表或授权的服务供应商 (ASP)。

Sun StorEdge SAM-FS 环境中所管理的各设备之间的关系，是在主配置文件 /etc/opt/SUNWsamfs/mcf 中定义的。 mcf 文件指定了 Sun StorEdge SAM-FS 环境中所包括的可移动介质设备、自动化库和文件系统。在 mcf 文件中，每一个设备均分配有唯一的设备标识。 mcf 文件中的条目还可定义手动安装的存档设备和自动化库目录文件。

系统将尽可能使用标准的 Solaris 磁盘和磁带设备驱动程序。对于 Solaris 操作系统 (OS) 不可直接支持的设备 （例如某些特定的库和光盘设备）， Sun StorEdge SAM-FS 软件包中提供了此类设备的专用驱动程序。

命令

Sun StorEdge QFS 和 Sun StorEdge SAM-FS 环境由文件系统、后台程序、进程、各种命令 （用户命令和管理员命令等）和工具组成。本节介绍 Sun StorEdge QFS 和 Sun StorEdge SAM-FS 软件销售套件中提供的命令。

Sun StorEdge QFS 和 Sun StorEdge SAM-FS 的命令可与标准 UNIX 文件系统命令配合使用。但其中有些命令仅可在某一种产品上使用。所有命令均收录在 UNIX man(1) 页中。

*《Sun StorEdge QFS 和 Sun StorEdge SAM-FS 文件系统管理指南》*中概要地介绍了后台程序，但是各个后台程序都会在本文档资料集的适当之处予以详细介绍。

本节介绍这些命令，并指出哪些命令可以在 Sun StorEdge QFS 或 Sun StorEdge SAM-FS 文件系统中使用。有关的详细信息，请参阅软件销售套件中提供的手册页。

本部分包括下列主题：

用户命令

默认情况下，文件系统操作对最终用户是透明的。但视您的站点具体情况而定，您可能想为站点的用户提供一些命令，以便可以更好地调整某些操作。表 1-2 中概要性地列出了这些命令。

表 1-2 用户命令

命令	说明	使用环境
archive(1)	将文件存档并设置文件存档属性。	Sun StorEdge SAM-FS
release(1)	释放磁盘空间并设置文件的释放属性。	Sun StorEdge SAM-FS
request(1)	创建可移动介质文件。	Sun StorEdge SAM-FS
sdu(1)	概要性地说明磁盘用途。sdu(1) 命令是基于 du(1) 命令的 GNU 版本。	Sun StorEdge QFS Sun StorEdge SAM-FS
segment(1)	设置分段文件属性。	Sun StorEdge SAM-FS
setfa(1)	设置文件属性。	Sun StorEdge QFS Sun StorEdge SAM-FS
sfind(1)	在目录层次结构中搜索文件。 sfind(1) 命令基于 find(1) 命令的 GNU 版本，并且包含基于 Sun StorEdge QFS 和 Sun StorEdge SAM-FS 文件属性的搜索选项。	Sun StorEdge QFS Sun StorEdge SAM-FS
sls(1)	列出目录的内容。 sls(1) 命令基于 ls(1) 命令的 GNU 版本，并且包含用于显示文件系统属性和信息的选项。	Sun StorEdge QFS Sun StorEdge SAM-FS

表 1-2　用户命令　*（接上页）*

命令	说明	使用环境
squota(1)	报告限额信息。	Sun StorEdge QFS Sun StorEdge SAM-FS
ssum(1)	设置文件校验和属性。	Sun StorEdge SAM-FS
stage(1)	设置文件登台属性并将脱机文件复制到磁盘。	Sun StorEdge SAM-FS

一般系统管理员命令

表 1-3 概要性地介绍了可用于维护和管理系统的命令。

表 1-3　一般系统管理员命令

命令	说明	使用环境
samcmd(1M)	执行一个 samu(1M) 操作员界面实用程序命令。	Sun StorEdge QFS Sun StorEdge SAM-FS
samd(1M)	开始或终止自动和可移动介质后台程序。	Sun StorEdge SAM-FS
samexplorer(1M)	生成 Sun StorEdge QFS 或 Sun StorEdge SAM-FS 诊断报告脚本	Sun StorEdge QFS Sun StorEdge SAM-FS
samset(1M)	更改 Sun StorEdge SAM-FS 设置。	Sun StorEdge SAM-FS
samu(1M)	调用全屏且基于文本的操作员界面。此界面基于 curses(3CURSES) 软件库。samu 实用程序用于显示设备状态，且操作员可通过它来控制自动化库。	Sun StorEdge QFS Sun StorEdge SAM-FS

文件系统命令

表 1-4 概要性地介绍了可用于维护文件系统的命令。

表 1-4 文件系统命令

命令	说明	使用环境
mount(1M)	安装文件系统。此命令的手册页名称为 mount_samfs(1M)。	Sun StorEdge QFS Sun StorEdge SAM-FS
qfsdump(1M) qfsrestore(1M)	创建或恢复转储文件，该文件包含与 Sun StorEdge QFS 文件系统相关联的文件数据和元数据。	Sun StorEdge QFS
sambcheck(1M)	列出文件系统块用法。	Sun StorEdge QFS Sun StorEdge SAM-FS
samchaid(1M)	更改文件管理集 ID 属性。与限额一起使用。	Sun StorEdge QFS Sun StorEdge SAM-FS
samfsck(1M)	检查和修复文件系统中的元数据冲突，并收回已分配但未使用的磁盘空间。	Sun StorEdge QFS Sun StorEdge SAM-FS
samfsconfig(1M)	显示配置信息。	Sun StorEdge QFS Sun StorEdge SAM-FS
samfsdump(1M) samfsrestore(1M)	创建或恢复转储文件，该文件包含与 Sun StorEdge SAM-FS 文件系统相关联的元数据。	Sun StorEdge SAM-FS Sun SAM-QFS
samfsinfo(1M)	显示有关 Sun StorEdge QFS 或 Sun StorEdge SAM-FS 文件系统布局的信息。	Sun StorEdge QFS Sun StorEdge SAM-FS
samfstyp(1M)	确定 Sun StorEdge QFS 或 Sun StorEdge SAM-FS 文件系统类型。	Sun StorEdge QFS Sun StorEdge SAM-FS
samgrowfs(1M)	通过添加磁盘设备来扩展文件系统。	Sun StorEdge QFS Sun StorEdge SAM-FS
sammkfs(1M)	在磁盘设备中初始化新的文件系统。	Sun StorEdge QFS Sun StorEdge SAM-FS
samncheck(1M)	如果提供安装点和索引节点编号，则返回完整的目录路径名。	Sun StorEdge QFS Sun StorEdge SAM-FS
samquota(1M)	报告、设置或重新设置限额信息。	Sun StorEdge QFS Sun StorEdge SAM-FS
samquotastat(1M)	有关激活的和未激活的文件系统限额的报告。	Sun StorEdge QFS Sun StorEdge SAM-FS

表 1-4　文件系统命令　*（接上页）*

命令	说明	使用环境
samsharefs(1M)	管理 Sun StorEdge QFS 共享文件系统配置信息。	Sun StorEdge QFS
samtrace(1M)	转储跟踪缓冲区。	Sun StorEdge QFS Sun StorEdge SAM-FS
samunhold(1M)	释放 SANergy 文件保持。	Sun StorEdge QFS Sun StorEdge SAM-FS
trace_rotate(1M)	循环更新跟踪文件。	Sun StorEdge QFS Sun StorEdge SAM-FS

自动化库命令

表 1-5 概要性地介绍可用于配置、初始化和维护 Sun StorEdge SAM-FS 环境中的自动化库和设备的自动化库命令。

表 1-5　自动化库命令

命令	说明
auditslot(1M)	核查指定自动化库中的单个介质卡盒端口。
build_cat(1M)	建立自动化库的介质目录文件。此外，它还可用于生成目录文件。
chmed(1M)	设置或清除特定卡盒上的库目录标记和数值。
cleandrive(1M)	请求载入附带清洁磁带的磁带驱动器。
dump_cat(1M)	以各种 ASCII 格式显示二进制目录文件的内容。
import(1M) samexport(1M)	通过将卡盒放入邮箱中来从库中导入或导出卡盒。对于通过网络连接的库，此命令用于更新库目录，而不是从物理上移动卡盒。
samload(1M) unload(1M)	载入或卸载指定设备的卡盒。
move(1M)	将卡盒从一个端口移动到另一个端口。
odlabel(1M)	标记光盘，以便用于 Sun StorEdge SAM-FS 系统。
samdev(1M)	添加 /dev/samst 逻辑设备条目。用于传递自动化库、光盘和磁带驱动器信息。
tplabel(1M)	标记磁带，以便用于 Sun StorEdge SAM-FS 系统。

存档程序命令

表 1-6 概要性地介绍 Sun StorEdge SAM-FS 环境中控制存档程序操作的命令。

表 1-6 存档程序命令

命令	说明
archiver(1M)	评估存档程序命令文件的语法完整性和语义准确性。
archiver.sh(1M)	记录异常的存档程序事件。
showqueue(1M)	显示存档程序队列文件的内容。
reserve(1M) unreserve(1M)	保留卷和取消保留卷。

专用维护命令

表 1-7 概要性地介绍可用于 Sun StorEdge SAM-FS 环境的各种维护命令。

表 1-7 专用维护命令

命令	说明
archive_audit(1M)	生成关于每个卡盒中所有已存档文件的报告。
dmpshm(1M)	转储共享的内存段。
exarchive(1M)	管理 （交换）存档副本。
itemize(1M)	编制光盘目录。
rearch(1M) unrearch(1M)	标记或取消标记要重新存档的存档条目。
sam-recycler(1M)	收回过期存档副本在存档介质中占用的空间。
sam-releaser(1M)	从在线磁盘高速缓存文件系统中释放磁盘空间。
samdev(1M)	在 /dev/samst 目录中创建符号链接，这个目录指向 Sun StorEdge SAM-FS 文件系统所要使用的实际设备。此命令的功能与 UNIX makedev(1M) 命令相似。
samset(1M)	更改或显示 Sun StorEdge SAM-FS 操作中所使用的变量。
set_admin(1M)	添加或删除管理员组执行管理员命令的权限。
set_state(1M)	设置 Sun StorEdge SAM-FS 设备的状态。
stageback.sh(1M)	从 Sun StorEdge SAM-FS 或 Sun SAM-QFS 存档磁带登台文件

表 1-7　专用维护命令　*（接上页）*

命令	说明
star(1M)	创建磁带存档，并且添加或抽取文件。此命令是 tar(1) 命令的 GNU 版本，且已扩展为可在 Sun StorEdge SAM-FS 文件系统中使用。如果在进行故障恢复时需要从存档磁带读取数据，那么可以使用这条命令。
tapealert(1M)	对 TapeAlert 事件进行解码。
unarchive(1M)	删除一个或多个文件的存档条目。
undamage(1M)	将一个或多个文件或目录的存档条目标记为 “未损坏”。

可在站点处自定义的脚本

表 1-8 概要性地介绍了可在站点处进行自定义的脚本，这些脚本可用于监视和控制 Sun StorEdge SAM-FS 环境。默认情况下，软件将这些脚本安装到 /opt/SUNWsamfs/examples 目录中。您可将这些脚本从 /opt/SUNWsamfs/examples 移动至 /etc/opt/SUNWsamfs/scripts，并修改它们以执行您的站点所需的操作。有关这些脚本的详细信息，请参阅各自的手册页。

表 1-8　可在站点处自定义的脚本

脚本	说明
dev_down.sh(1M)	当设备被标记为 down （禁用）或 off （关闭）时，向 root 用户发送电子邮件。
load_notify.sh(1M)	当 Sun StorEdge SAM-FS 软件请求不在库内的卡盒时，向操作员发送通知。
log_rotate.sh(1M)	循环更新日志文件。
recover.sh(1M)	恢复自上次执行 samfsdump(1M) 后存档的文件。
restore.sh(1M)	将文件恢复为在线状态或部分在线状态。
stageback.sh(1M)	从存档介质登台文件。
tarback.sh(1M)	从存档介质重新载入文件。

应用程序编程接口

可使用应用程序编程接口 (API) 来从用户应用程序中发出文件系统请求。用户可以从本地或远程向运行有此文件系统的计算机发送请求。 API 包括 `libsam` 和 `libsamrpc` 库。这些库包括一些库例程，可用于获取文件状态，设置文件的存档、释放和登台属性以及控制自动化库的库目录。 `sam-rpcd` 远程过程调用后台程序对远程请求进行处理。要自动地启动 `sam-rpcd` 后台程序，可在 `defaults.conf` 文件中设置 `samrpc=on`。

有关 API 的详细信息，请参阅 `intro_libsam`(3) 手册页。此手册页概要地介绍了如何使用 `libsam` 和 `libsamrpc` 中的库例程。

可操作实用程序

在 Sun StorEdge SAM-FS 环境中，可使用 `samu`(1M) 操作员实用程序以及 SAM-QFS Manager 来执行基本操作。表 1-9 概要性地介绍了这些可操作工具。

表 1-9 可操作工具

GUI 工具	说明
SAM-QFS Manager	为 Sun StorEdge QFS 和 Sun StorEdge SAM-FS 软件提供基于 Web 的图形用户界面。可通过这个界面来配置、控制、监视和重新配置 Sun StorEdge QFS 和 Sun StorEdge SAM-FS 环境中的组件。有关如何安装 SAM-QFS Manager 的信息，请参阅 *《Sun StorEdge QFS 和 Sun StorEdge SAM-FS 软件安装和配置指南》*。有关如何使用 SAM-QFS Manager 的信息，请参阅其在线帮助。
`samu`(1M)	提供用于访问 `samu`(1M) 操作员实用程序的起始点。

第二章

在 Sun StorEdge SAM-FS 环境中使用自动化库和手动载入的驱动器

自动化库 是一种自动控制的设备，它可在无操作人员参与的情况下，自动载入和卸载可移动卡盒。卡盒可以导入自动化库或从中导出。系统能够自动载入或卸载卡盒。存档和登台进程根据站点定义的方案来分配要使用的驱动器数量。自动化库也可称为介质更换器、自动光盘存储器、传输器、资料库或介质库。

以下几节从各个方面介绍了如何在 Sun StorEdge SAM-FS 环境中使用自动化库。*《Sun StorEdge QFS 和 Sun StorEdge SAM-FS 软件安装和配置指南》*中提供了初始配置说明，本章将介绍自动化库和手动载入的驱动器的操作说明。此外，本章还介绍在被请求的卷不在库中时，用于提示操作员载入该卷的通知工具。

注意 – Sun StorEdge SAM-FS 软件可与许多厂商的自动化库进行交互。有关自动化库型号、固件级别以及其他兼容性方面的信息，请与 Sun 的客户支持部门联系。

某些自动化库的功能所产生的操作可能不同于本章介绍的操作。要确定对于 Sun StorEdge SAM-FS 环境中所使用的自动化库，其供应商是否提供有额外的操作说明，请参阅第 35 页的 “特定供应商库的基本操作”。

本章包括以下主题：

约定

执行本章所述的基本操作过程时，通常需要使用 samcmd(1M) 命令、 samu(1M) 操作员实用程序以及下列命令：

- tplabel(1M)
- odlabel(1M)
- auditslot(1M)
- cleandrive(1M)
- chmed(1M)
- import(1M)
- set_state(1M)
- samexport(1M)

但是，在许多情况下，您可以选用多种方法来执行所述的任务。您可以在 SAM-QFS Manager 中执行其中许多任务，这个管理器是 Sun StorEdge QFS 和 Sun StorEdge SAM-FS 软件基于 Web 的图形用户界面 (GUI)。可通过这个界面来配置、控制、监视和重新配置 Sun StorEdge QFS 和 Sun StorEdge SAM-FS 环境中的组件。有关如何安装 SAM-QFS Manager 的信息，请参阅 *《Sun StorEdge QFS 和 Sun StorEdge SAM-FS 软件安装和配置指南》*。有关如何使用 SAM-QFS Manager 的信息，请参阅其在线帮助。

命令变量

许多命令接受一组通用的变量。表 2-1 列出了这些变量。

表 2-1　命令变量

变量	含义
eq	所访问的设备在 mcf 文件中定义的设备序号。可以识别的设备包括自动化库、驱动器或文件系统。
slot	自动化库中存储端口的编号，与库目录中标识的编号相同。
partition	磁光盘的一面。 partition 必须为 1 或 2。
media_type	介质类型。有关有效介质类型的列表，请参阅 mcf(4) 手册页。
vsn	分配给卷的卷序列名。

某些命令可以接受多种变量组合，这视您的环境而定。例如，在 samu(1M) 操作员实用程序中， load 命令具有以下两种格式：

```
:load eq:slot
:load media_type.vsn
```

注意以下细节：

- 第一种格式使用冒号 (:) 分隔 *eq* 和 *slot*。
- 第二种格式使用句点 (.) 分隔 *media_type* 和 *vsn*。

术语

本章所使用的某些术语对您来说可能是新术语，因此，表 2-2 中列出了一些常用术语及其含义。

表 2-2　术语

术语	含义
自动化库	用于存储磁带或光盘卡盒的自动设备。
卡盒	磁带或磁光盘卡盒。磁光盘卡盒可以包含一个或多个卷或分区。
分区	整个磁带或磁光盘的一面。一个分区只能包含一个卷。
卷	卡盒中用于存储数据的命名区域。一个卡盒中可以有一个或多个卷。双面卡盒有两个卷，每一面为一个卷。系统采用卷序列名 (VSN) 来标识卷。

自动化库操作

有几项基本操作在所有自动化库中基本上都是相同的。本节介绍了以下基本操作：

- 第 17 页的 “启动可移动介质的操作”
- 第 16 页的 “停止可移动介质的操作”
- 第 17 页的 “打开自动化库”
- 第 17 页的 “关闭自动化库”
- 第 18 页的 “将卡盒载入自动化库”
- 第 19 页的 “从驱动器中卸载卡盒”

- 第 19 页的 “标记卡盒”
- 第 21 页的 “核查卷”
- 第 22 页的 “核查自动化库 （仅限于直接连接）”
- 第 22 页的 “使用清洁卡盒”
- 第 24 页的 “清洁磁带驱动器”
- 第 25 页的 “清除介质错误”
- 第 26 页的 “从驱动器中取出卡住的卡盒”
- 第 27 页的 “目录操作和卡盒的导入与导出”
- 第 32 页的 “启用载入通知”

▼ 停止可移动介质的操作

可以停止可移动介质的操作，并保持 Sun StorEdge SAM-FS 系统的安装状态。例如，在您希望手动操作库中的卡盒时，可能需要执行此操作。恢复介质操作之后，未完成的登台操作将被重新执行，而存档操作也将随之恢复。

● **要停止可移动介质的操作，可使用** `samcmd`**(1M)** `idle` **和** `samd`**(1M)** `stop` **命令。**

这些命令的使用格式如下：

```
samcmd idle eq
samd stop
```

其中的 *eq*，用于输入所访问的设备在 `mcf` 文件中定义的设备序号。要将驱动器置于空闲状态，请对 `mcf` 文件中配置的每一个 *eq* 运行 `samcmd idle` *eq* 命令。

也可以使用 `samu`(1M) 操作员实用程序或使用 SAM-QFS Manager 将驱动器置于空闲状态。

注意 – 在运行 `samd`(1M) `stop` 命令之前， Sun StorEdge SAM-FS 环境中的驱动器应该处于空闲状态。其目的是使存档程序、登台程序和其它进程结束当前的任务。如果未能成功运行 `samd`(1M) `stop` 命令，则在恢复存档、登台或其它活动时可能导致意外的结果。

▼ 启动可移动介质的操作

通常，当安装了 Sun StorEdge SAM-FS 文件系统后，可移动介质的操作便已启动。

- **要手动启动可移动介质操作，而不安装文件系统，请输入 `samd`(1M) `start` 命令。**
  此命令的使用格式如下：

```
# samd start
```

如果在输入上述命令时可移动介质操作已在运行，则会生成以下消息：

```
SAM-FS sam-amld daemon already running
```

有关 `samd`(1M) 命令的详细信息，请参阅 `samd`(1M) 手册页。

▼ 打开自动化库

当某个库处于 `on`（打开）状态时，那么它已经处于 Sun StorEdge SAM-FS 系统的控制之下，且可以继续执行一般性的操作。打开库时， Sun StorEdge SAM-FS 软件会执行以下操作：

- 根据设备的内部状态来查询设备。它可以发现磁带的位置、是否使用了条码等。
- 更新目录和其它内部结构。

- **使用 `samcmd`(1M) `on` 命令打开自动化库。**
  此命令的使用格式如下：

```
samcmd on eq
```

其中的 *eq*，用于指定所访问的自动化库在 `mcf` 文件中定义的设备序号。

另外，还可使用 `samu`(1M) 或 SAM-QFS Manager 执行此任务。

▼ 关闭自动化库

将自动化库置于 `off`（关闭）状态可以停止 I/O 操作，并使该库不再受 Sun StorEdge SAM-FS 控制。此时，卡盒将不能自动移动。请注意，自动化库中的驱动器仍处于 `on`（打开）状态。若要执行以下任务，可能需要关闭自动化库：

- 仅停止 Sun StorEdge SAM-FS 自动化库的操作。
- 关闭自动化库的电源。

● **使用 samcmd(1M) off 命令关闭自动化库。**

此命令的使用格式如下：

```
samcmd off eq
```

其中的 *eq*，用于指定所访问的自动化库在 mcf 文件中定义的设备序号。

另外，还可使用 samu(1M) 或 SAM-QFS Manager 执行此任务。

▼ 将卡盒载入自动化库

当请求某个 VSN 以进行存档或登台操作时，系统会自动将卡盒载入驱动器。载入是指将卡盒从存储端口移入驱动器，并使之处于就绪状态的过程。

● **使用 samcmd(1M) load 命令手动载入卡盒。**

即使驱动器处于 unavail （不可用）状态，也可使用此命令。此命令具有以下两种格式：

```
samcmd load eq:slot[:partition]
samcmd load media_type.vsn
```

表 2-3　samcmd(1M) load 的变量

变量	含义
eq	所访问的驱动器在 mcf 文件中定义的设备序号。
slot	自动化库中存储端口的编号，与库目录中标识的编号相同。
media_type	介质类型。有关有效介质类型的列表，请参阅 mcf(4) 手册页。
partition	磁光盘的一面。 partition 必须为 1 或 2。此变量不适用于磁带卡盒。
vsn	分配给卷的卷序列名。

另外，还可使用 samu(1M) 或 SAM-QFS Manager 执行此任务。

当手动载入卡盒时，它通常会载入库中的下一个可用驱动器。如果不希望某个驱动器这样自动载入，可使用 samu(1M) 实用程序的 :unavail 命令，或使用 SAM-QFS Manager 更改该设备的状态。例如，在故障恢复操作或对磁带进行分析期间，就可能会执行此操作。

▼ 从驱动器中卸载卡盒

当不再需要使用卷时，系统会自动卸载卡盒。也可手动卸载驱动器。卸载是指从驱动器中取出卡盒。

- **使用 samcmd(1M) unload 命令手动卸载卡盒。**

  即使驱动器处于 unavail （不可用）状态，也可使用此命令。此命令的使用格式如下：

```
samcmd unload eq
```

其中的 *eq*，用于指定所访问的自动化库在 mcf 文件中定义的设备序号。

另外，还可使用 samu(1M) 或 SAM-QFS Manager 执行此任务。

标记卡盒

标记卡盒的过程取决于您是标记磁带还是标记光盘卡盒。以下两小节介绍了这些过程。

警告 – 标记和重新标记卡盒可使任何软件都无法再访问卡盒中当前存储的数据。应该仅在确定卡盒中存储的数据不再有用时，才重新标记卡盒。

▼ 标记或重新标记磁带

下面的 tplabel(1M) 命令行格式显示了在标记或重新标记磁带时最常用的选项：

```
tplabel [ -new | -old vsn ] -vsn vsn eq:slot
```

表 2-4 tplabel(1M) 的变量

变量	含义
vsn	卷序列名。如果是重新标记磁带，新 VSN 名可以与旧 VSN 名相同。
eq	所访问的自动化库或手动载入的驱动器在 mcf 文件中定义的设备序号。
slot	自动化库中存储端口的编号，与库目录中标识的编号相同。此变量不适用于手动载入的驱动器。

- **若要标记新磁带，应使用 `tplabel`(1M) 命令。**

  此命令的使用格式如下：

```
tplabel -new -vsn vsn eq:slot
```

- **若要重新标记现有磁带，应使用 `tplabel`(1M) 命令。**

  此命令的使用格式如下：

```
tplabel -old vsn -new -vsn vsn eq:slot
```

发出上述用以标记或重新标记磁带的命令后，系统会载入磁带并确定其位置，然后写入磁带标签。有关 `tplabel`(1M) 命令的详细信息，请参阅 `tplabel`(1M) 手册页。

另外，还可使用 SAM-QFS Manager 执行此任务。

▼ 标记或重新标记光盘

下面的 `odlabel`(1M) 命令行格式显示了在标记或重新标记光盘时最常用的选项：

```
odlabel [ -new | -old vsn ] -vsn vsn eq:slot:partition
```

表 2-5　`odlabel`(1M) 的变量

变量	含义
vsn	卷序列名。如果是进行重新标记，新 VSN 名可以与旧 VSN 名相同。
eq	所访问的自动化库或手动载入的驱动器在 `mcf` 文件中定义的设备序号。
slot	自动化库中存储端口的编号，与库目录中标识的编号相同。此变量不适用于手动载入的驱动器。
partition	磁光盘的一面。 partition 必须为 `1` 或 `2`。此变量不适用于磁带卡盒。

- **若要标记新光盘，应使用 `odlabel`(1M) 命令。**

  此命令的使用格式如下：

```
odlabel -new -vsn vsn eq:slot:partition
```

- **若要重新标记现有光盘，应使用 `odlabel`(1M) 命令。**

  此命令的使用格式如下：

```
odlabel -old vsn -vsn vsn eq:slot:partition
```

  发出上述用以标记或重新标记光盘的命令后，系统会载入光盘并确定其位置，然后写入光盘标签。有关 `odlabel`(1M) 命令的详细信息，请参阅 `odlabel`(1M) 手册页。

  另外，还可使用 SAM-QFS Manager 执行此任务。

▼ 核查卷

有时，需要在库目录中更新以前所报告的磁带或光盘卡盒上的剩余空间。`auditslot`(1M) 命令用于载入包含卷的卡盒、读取标签并更新存储端口的库目录条目。

- **使用 `auditslot`(1M) 命令核查卷。**

  此命令的使用格式如下：

```
auditslot [-e] eq:slot[:partition]
```

表 2-6 `auditslot`(1M) 的变量

变量	含义
`-e`	如果指定了 `-e` 选项，且介质为磁带，那么将会更新剩余的空间。否则，将不会进行更新。
eq	所访问的自动化库或手动载入的驱动器在 `mcf` 文件中定义的设备序号。
slot	自动化库中存储端口的编号，与库目录中标识的编号相同。此变量不适用于手动载入的驱动器。
partition	磁光盘的一面。 partition 必须为 `1` 或 `2`。此变量不适用于磁带卡盒。

有关 `auditslot`(1M) 命令的详细信息，请参阅 `auditslot`(1M) 手册页。

也可以使用 `samu`(1M) 实用程序的 `:audit` 命令或使用 SAM-QFS Manager 执行此任务。

▼ 核查自动化库 （仅限于直接连接）

注意 – 此任务不能在通过网络连接的自动化库上执行。

执行全面核查时，会将每一个卡盒载入驱动器、读取标签并更新库目录。在下列情况下应核查库：

- 移动了自动化库中的卡盒，但没有使用 Sun StorEdge SAM-FS 的命令。
- 如果怀疑库目录的状态有问题，且希望更新库目录 （例如在意外断电后）。
- 如果在没有配备邮箱的自动化库中添加、取出或移动了卡盒。

● **可使用 `samcmd`(1M) `audit` 命令执行自动化库的全面核查。**

此命令的使用格式如下：

```
samcmd audit eq
```

其中的 *eq*，用于指定所访问的自动化库在 `mcf` 文件中定义的设备序号。

也可以使用 `samu`(1M) 实用程序的 `:audit` 命令或使用 SAM-QFS Manager 执行此任务。

使用清洁卡盒

Sun StorEdge SAM-FS 系统允许您导入清洁卡盒来清洁磁带驱动器。此过程的具体步骤取决于清洁卡盒是否编有条码。以下几节介绍了使用清洁卡盒的不同情况。

清洁方法因制造商不同而不同。如果遇到这方面的问题，请参阅第 35 页的 “特定供应商库的基本操作”，以确定是否需要为您的设备执行特殊步骤。

注意 – 此任务不能在通过网络连接的自动化库上执行。

▼ 重设清洁循环次数

清洁磁带仅在一定的清洁循环次数内有效。可以使用 `samu`(1M) 实用程序的 `:v` 命令或使用 SAM-QFS Manager 来显示剩余的清洁次数。

Sun StorEdge SAM-FS 系统可以跟踪每个清洁磁带的清洁循环次数，并在剩余循环次数等于零时弹出磁带。例如， DLT 清洁磁带的循环次数为 20，而 Exabyte 清洁磁带的循环次数则为 10。每次导入清洁磁带时，系统均会自动将清洁循环次数重设为该类型磁带的最高循环次数。

如果系统支持自动清洁功能，但自动化库中所有清洁磁带的剩余循环次数均为零，则系统会将驱动器设置为 off（关闭）状态，并在 Sun StorEdge SAM-FS 日志中记录一条消息。

- **可使用 chmed(1M) 命令将清洁磁带的循环次数重设为零。**

  此命令的使用格式如下：

```
chmed -count count media_type.vsn
```

表 2-7　chmed(1M) 的变量

变量	含义
count	您为清洁磁带重设的清洁循环次数。
media_type	介质类型。有关有效介质类型的列表，请参阅 mcf(4) 手册页。
vsn	分配给卷的卷序列名。

▼ 使用具有条码的清洁卡盒

如果清洁卡盒编有条码，则可使用 import(1M) 命令导入清洁卡盒。

1. **确保清洁卡盒的条码为 CLEAN，或以字母 CLN 开头。**
2. **使用 import(1M) 命令导入清洁卡盒。**

   此命令的使用格式如下：

```
import eq
```

其中的 *eq*，用于指定所访问的自动化库在 mcf 文件中定义的设备序号。

Sun StorEdge SAM-FS 系统会将卡盒从邮箱移至存储端口，并更新每个卡盒的库目录。另外，发出此命令后，系统将设置清洁介质标记，并根据介质的类型将访问计数设置为适当的清洁循环次数。每使用介质清洁一次驱动器，访问计数便减少一次。

例如，下面的命令是将一个在 mcf 文件中编号为 50 的清洁磁带导入自动化库：

```
# import 50
```

另外，还可使用 samu(1M) 或 SAM-QFS Manager 执行此任务。

▼ 使用没有条码的清洁卡盒

如果清洁卡盒没有条码，则必须首先将其导入。此时，它未被标记为清洁卡盒。请执行以下步骤：

1. **使用 `import`(1M) 命令导入卡盒。**

   此命令的使用格式如下：

```
import eq
```

   其中的 *eq*，用于指定所访问的自动化库在 `mcf` 文件中定义的设备序号。

2. **使用 `chmed`(1M) 命令将设备类型更改为清洁卡盒。**

   您必须知道自动化库的设备序号以及清洁卡盒被载入的存储端口。

   在下面的命令行示例中，自动化库的设备序号为 50，且清洁卡盒位于存储端口 77 中：

```
# chmed +C 50:77
```

   上述命令将卡盒类型更改为清洁卡盒类型。

3. **再次使用 `chmed`(1M) 命令，设置清洁循环次数。**

   下面的命令示例为上一步骤中的清洁卡盒设置了清洁循环次数：

```
# chmed -count 20 50:77
```

   有关 `chmed`(1M) 命令的详细信息，请参阅 `chmed`(1M) 手册页。

▼ 清洁磁带驱动器

注意 – Sun StorEdge SAM-FS 系统不支持自动地清洁通过网络连接的库。您应使用供应商提供的库管理软件来自动清洁此类库。

如果硬件支持清洁磁带的使用，那么 Sun StorEdge SAM-FS 环境也支持使用清洁磁带。如果某个磁带驱动器要求清洁，系统会自动载入清洁磁带。

如果您的系统使用编有条码的标签，则在条码标签中，清洁磁带的 VSN 必须为 `CLEAN` 或以字母 `CLN` 开头。另外，可以使用 `chmed`(1M) 命令将 VSN 标记为清洁磁带并设置清洁循环计数。系统允许安装多个清洁磁带。

注意 – 某些驱动器错误可能会导致重复载入清洁卡盒，直到执行完所有的清洁循环。为防止发生此类情况，您可以使用 chmed(1M) 命令限制清洁卡盒的清洁循环次数。例如：

```
# chmed -count 20 50:77
```

如果不能进行自动清洁，而且系统使用条码，那么可手动执行以下步骤来请求清洁驱动器：

- **使用 cleandrive(1M) 命令。**

  此命令的使用格式如下：

```
cleandrive eq
```

  其中的 *eq*，用于指定所访问的自动化库在 mcf 文件中定义的设备序号。这是要载入清洁卡盒的驱动器。

▼ 清除介质错误

当卡盒发生硬件或软件错误时， Sun StorEdge SAM-FS 系统会在 VSN 目录中设置 media error （介质错误）标记。对于任何生成 media error 信号的给定卡盒，均可使用 chmed(1M) 命令清除其错误，然后尝试使用该卡盒。可使用 samu(1M) 实用程序的 v 命令或 SAM-QFS Manager 显示 media error 标记。

1. **使用 chmed(1M) 命令清除 media error 标记。**

   按以下格式使用此命令清除 media error 标记：

```
chmed -E media_type.vsn
```

表 2-8 chmed(1M) 的变量

变量	含义
media_type	介质类型。有关有效介质类型的列表，请参阅 mcf(4) 手册页。
vsn	分配给卷的卷序列名。

2. 运行 auditslot(1M) 命令更新剩余空间信息。

此命令的使用格式如下：

```
auditslot -e eq:slot[:partition]
```

表 2-9 auditslot(1M) 的变量

变量	含义
-e	如果指定了 -e 选项，且介质为磁带，那么将会更新剩余的空间。否则，将不会进行更新。
eq	所访问的自动化库或手动载入的驱动器在 mcf 文件中定义的设备序号。
slot	自动化库中存储端口的编号，与库目录中标识的编号相同。此变量不适用于手动载入的驱动器。
partition	磁光盘的一面。 partition 必须为 1 或 2。此变量不适用于磁带卡盒。

有关 auditslot(1M) 命令的详细信息，请参阅 auditslot(1M) 手册页。

也可以使用 samu(1M) 实用程序的 :audit 命令或使用 SAM-QFS Manager 执行此任务。

▼ 从驱动器中取出卡住的卡盒

如果卡盒卡在驱动器中，请执行以下步骤。

1. 使用 samcmd(1M) off 命令关闭自动化库中的驱动器。

此命令的使用格式如下：

```
samcmd off eq
```

其中的 *eq*，用于指定所访问的驱动器在 mcf 文件中定义的设备序号。

另外，还可使用 samu(1M) 或 SAM-QFS Manager 执行此任务。

2. 使用 samcmd(1M) off 命令关闭自动化库。

此命令的使用格式如下：

```
samcmd off eq
```

其中的 *eq*，用于指定所访问的库在 mcf 文件中定义的设备序号。

另外，还可使用 samu(1M) 或 SAM-QFS Manager 执行此任务。

3. **从驱动器中物理取出卡盒。**

   确保不要损坏卡盒或驱动器。

4. **使用 samcmd(1M) on 命令打开自动化库和驱动器。**

   为驱动器和库分别运行一条此命令。此命令的使用格式如下：

```
samcmd on eq
```

   其中的 *eq*，用于指定所访问的自动化库或驱动器在 mcf 文件中定义的设备序号。

   如果自动化库在打开后执行了核查，则本过程结束。如果未执行核查，请执行下一步骤。

5. **如果将卡盒放回其存储端口，请使用 chmed(1M) 命令调整库目录，以便为损坏的磁带设置占用标记。**

   此命令的使用格式如下：

```
chmed +o eq:slot
```

表 2-10 chmed(1M) 的变量

变量	含义
eq	所访问的自动化库或驱动器在 mcf 文件中定义的设备序号。
slot	自动化库中存储端口的编号，与库目录中标识的编号相同。此变量不适用于手动载入的驱动器。

有关 chmed(1M) 命令的详细信息，请参阅 chmed(1M) 手册页。

如果您取出卡盒，并希望在以后将其放回，则必须将卡盒导入自动化库。

目录操作和卡盒的导入与导出

在自动化库中物理地添加 （导入）和从中取出 （导出）卡盒，使您可以执行多项功能，这其中包括：

- 更换卡盒。
- 将卡盒重新定位为离站存储，以备今后进行故障恢复时使用。如果要执行此任务，可使用 chmed(1M) 命令的 -I 选项，来指定诸如卡盒存储位置等其它信息。

导入和导出卡盒时，还可更新库目录。在 Sun StorEdge SAM-FS 系统中，可使用 `import`(1M) 与 `samexport`(1M) 命令完成此任务。另外，还可使用 SAM-QFS Manager 执行这些任务。

库*目录*是一个中央仓库，其中包含了 Sun StorEdge SAM-FS 环境在查找自动化库中的卡盒时所需的所有信息。库目录文件是一个二进制 UFS 驻留文件，其中包括自动化库中每一个端口的有关信息。该文件中的信息包括：与端口中的卡盒相关联的一个或多个卷序列名 (VSN)；该卡盒的容量和剩余空间；以及指示卡盒只读、写保护、回收和其它状态信息的标记。

Sun StorEdge SAM-FS 环境处理目录的方式不尽相同，这取决于自动化库连接到服务器的方式，具体如下：

- 如果自动化库采用直接连接方式，则库目录的条目与自动化库中的物理端口之间是一一对应关系。库目录中的第一个条目是自动化库中的第一个端口。需要某个卡盒时，系统将首先查询库目录，以确定哪一个端口包含该 VSN，然后发出命令以将该端口中的卡盒载入驱动器。
- 如果自动化库采用网络连接方式，则库目录中的条目与自动化库中的端口不是直接的对应关系。它是自动化库中已知 VSN 的列表。当请求某个卡盒时，系统将向供应商的软件发送请求以将该 VSN 载入驱动器。供应商的软件可以确定包含该 VSN 的存储端口。

每个自动化库处理卡盒导入和导出操作的方式，都因系统特性和供应商提供的软件不同而存在差异。例如，对于 ACL 4/52 库，您需要先运行 `move` 命令将卡盒移至导入 / 导出设备中，然后才能从自动化库中导出卡盒。

注意 – 通过网络连接的自动化库则使用它们自己的实用程序来导入和导出卡盒，因此 `import`(1M) 和 `samexport`(1M) 命令只更新 Sun StorEdge SAM-FS 系统使用的库目录条目。如果您使用的是通过网络连接的自动化库，请参阅第 35 页的 “特定供应商库的基本操作”，以了解有关导入和导出卡盒的信息。

跟踪导出的介质 — 历史记录

Sun StorEdge SAM-FS 历史记录可以跟踪记录从自动化库或手动安装的设备中导出的卡盒。历史记录类似于一个虚拟库，但它没有已定义的硬件设备。与自动化库一样，它也在 `mcf` 文件中进行配置；具有用于记录与其关联的所有卡盒的目录；可以导入和导出卡盒；而且在 SAM-QFS Manager 中显示为另一自动化库。

可以在 `mcf` 文件中，使用 `hy` 设备类型来配置历史记录。如果没有在 `mcf` 文件中配置历史记录，它将按以下方式创建：

```
historian  n+1  hy  -  on  /var/opt/SUNWsamfs/catalog/historian
```

在以上条目中，*n+1* 是 mcf 文件中的最后一个设备序号加 1。如果希望目录使用不同的设备序号或路径名，则只需在 mcf 中对历史记录进行定义。

历史记录第一次启动时，历史记录库目录将被初始化为具有 32 个条目。请确保文件系统中的目录足以容纳整个目录。您可能希望跟踪已从库中导出的现有 Sun StorEdge SAM-FS 卡盒。若是如此，您需要按 build_cat(1M) 手册页中的说明，根据现有的卡盒建立历史记录目录。

defaults.conf 文件中的以下两个配置指令将影响历史记录的操作：

- 如果使用了 exported_media = unavailable 指令，则从自动化库中导出的任何卡盒均会被标记为不能用于历史记录。请求此类卡盒将产生 EIO 错误。
- 如果使用了 attended = no 指令，它将通知历史记录无操作员处理载入请求。如果已通知历史记录载入卡盒，但实际并未载入，则会产生 EIO 错误。

有关配置的详细信息，请参阅 historian(7) 和 defaults.conf(4) 手册页。

对自动化库执行导入和导出操作

邮箱 是自动化库中的一个区域，它用于在自动化库中添加和取出卡盒。import(1M) 命令用于将卡盒从邮箱移至存储端口。 samexport(1M) 命令用于将卡盒从存储端口移至邮箱。对于大多数库，如果在 Sun StorEdge SAM-FS 软件启动时，邮箱中含有卡盒，则软件会在启动期间自动导入此卡盒。

导入和导出的具体操作过程因制造商而异。如果遇到这方面的问题，请参阅第 35 页的 “特定供应商库的基本操作”，以确定是否需要为您的设备执行特殊步骤。

以下几节介绍如何导入和导出卡盒：

- 第 30 页的 “向使用邮箱的库中导入卡盒”
- 第 30 页的 “从使用邮箱的库中导出卡盒”
- 第 31 页的 “向不使用邮箱的库中导入卡盒”
- 第 32 页的 “从不使用邮箱的库中导出卡盒”

▼ 向使用邮箱的库中导入卡盒

要将卡盒导入使用邮箱的自动化库中，请执行以下步骤。

1. **按制造商建议的操作步骤打开邮箱。**

   邮箱的旁边通常配有一个按钮。有时，邮箱可能只配有一个端口 （在供应商的文档中称为*邮箱端口*）。

2. **将卡盒手动放入邮箱。**

3. **关闭邮箱。**

4. **使用 `import`(1M) 命令导入卡盒。**

   此命令的使用格式如下：

   ```
   import eq
   ```

   其中的 *eq*，用于指定所访问的库在 `mcf` 文件中定义的设备序号。

   系统将卡盒从邮箱移至存储端口，并更新每个卡盒的库目录。

   另外，还可使用 `samu`(1M) 或 SAM-QFS Manager 执行此任务。

▼ 从使用邮箱的库中导出卡盒

本过程用于将卡盒从存储端口移至邮箱或邮箱端口。要从使用邮箱的自动化库中导出 （弹出）卡盒，请执行以下步骤。

1. **使用 samexport(1M) 命令将卡盒从存储端口移至邮箱。**

   按以下某一种格式使用此命令：

```
samexport eq:slot
samexport media_type.vsn
```

表 2-11　samexport(1M) 的变量

变量	含义
eq	所访问的自动化库在 mcf 文件中定义的设备序号。
slot	自动化库中存储端口的编号，与库目录中标识的编号相同。
media_type	介质类型。有关有效介质类型的列表，请参阅 mcf(4) 手册页。
vsn	分配给卷的卷序列名。

   另外，还可使用 samu(1M) 或 SAM-QFS Manager 执行此步骤。

2. **按制造商建议的操作步骤打开邮箱或邮箱端口。**

   邮箱的旁边通常配有一个按钮。

▼ 向不使用邮箱的库中导入卡盒

1. **使用 samcmd(1M) unload 命令。**

   此命令的使用格式如下：

```
samcmd unload eq
```

   其中的 *eq*，用于指定所访问的库在 mcf 文件中定义的设备序号。

   等待系统完成其当前任务，将自动化库的状态设置为 off（关闭），然后将当前活动的目录转送至历史记录。

2. **解除自动化库挡门的锁定，打开挡门。**

3. **将卡盒载入可用的端口。**

4. **关闭自动化库的挡门，锁定挡门。**

   自动化库将重新初始化并扫描库中的卡盒。 Sun StorEdge SAM-FS 软件将所导入的卡盒的 VSN 添加到库目录中，从而更新该目录。此时，自动化库的状态会设置为 on （打开）。

▼ 从不使用邮箱的库中导出卡盒

1. 使用 samcmd(1M) unload 命令。

   此命令的使用格式如下：

   ```
   samcmd unload eq
   ```

   其中的 *eq*，用于指定所访问的库在 mcf 文件中定义的设备序号。

   等待系统完成其当前任务，将自动化库的状态设置为 off （关闭），然后将当前活动的目录转送至历史记录。

2. 解除自动化库挡门的锁定，打开挡门。

3. 从各个端口中取出所需的卡盒。

4. 关闭自动化库的挡门，锁定挡门。

   自动化库将重新初始化并扫描自动化库中的卡盒。系统使用库端口中当前卡盒的 VSN 来更新库目录。已取出的卡盒的 VSN 会从库目录中删除，并且只保留在历史记录文件中。此时，自动化库的状态会设置为 on （打开）。

▼ 启用载入通知

Sun StorEdge SAM-FS 软件会定期请求载入卡盒，以满足存档和登台需求。如果被请求的卡盒位于库中，则系统会自动处理该请求。如果被请求的卡盒不在库中，则需要操作员载入卡盒。如果已启用载入通知，则当需要获取不在库中的卡盒时，load_notify.sh(1M) 脚本会向有关人员发送电子邮件。

1. 成为超级用户。

2. 使用 cp(1) 命令，将载入通知脚本从其安装位置复制到可操作的位置。

   例如：

   ```
   # cp /opt/SUNWsamfs/examples/load_notify.sh
   /etc/opt/SUNWsamfs/scripts/load_notify.sh
   ```

3. 使用 more(1) 或其他命令检查 defaults.conf 文件。

   确保此文件中包含以下指令：

   - exported_media=available

■ attended=yes

默认情况下，系统已设置了这些指令。若要启用载入通知功能，请确保未更改这些指令。

4. **修改 load_notify.sh 脚本以便将通知发送给操作员。**

默认情况下，此脚本将向 root 用户发送电子邮件。不过，您可以编辑此脚本以将电子邮件发送给其他人员、拨打寻呼机，或提供其它通知方式。

手动载入驱动器操作

本节介绍在您配有手动载入的独立驱动器 （而不是自动化库）时所执行的操作。每一个手动载入的驱动器都有自己的单端口库目录。

▼ 载入卡盒

● **要将卡盒载入手动载入设备，请根据制造商的说明将卡盒放入驱动器。**

Sun StorEdge SAM-FS 系统会识别已载入的卡盒，读取标签并更新手册 （即单端口目录）。无需进行其它操作。

▼ 卸载卡盒

● **使用 samcmd(1M) idle 命令将驱动器置于空闲状态。**

此命令可以确保无任何存档或登台进程处于活动状态。此命令的使用格式如下：

```
samcmd idle eq
```

其中的 *eq*，用于指定所访问的驱动器在 mcf 文件中定义的设备序号。

完成所有 I/O 活动后，驱动器的状态将从 idle （空闲）切换为 off （关闭），然后弹出磁带。

如果是磁带，则先进行倒带，然后才能取出卡盒。如果是光盘卡盒，则会自动弹出。有关如何取出特定的卡盒，请参阅制造商提供的说明。

另外，还可使用 samu(1M) 或 SAM-QFS Manager 执行此任务。

▼ 查看库目录

- **使用 `samu`(1M) 实用程序的 `:v` 命令。**
  此命令的使用格式如下：

```
:v eq
```

  其中的 *eq*，用于指定所访问的库在 `mcf` 文件中定义的设备序号。

第三章

特定供应商库的基本操作

Sun StorEdge SAM-FS 环境中可以包含来自众多不同厂商的库。对于其中大多数库，您应该按第 13 页的 “在 Sun StorEdge SAM-FS 环境中使用自动化库和手动载入的驱动器”中所述步骤进行操作。但是，某些库需要执行供应商提供的特定操作步骤，本章将介绍这些步骤。

注意 – Sun StorEdge SAM-FS 软件可与许多厂商的自动化库兼容。有关自动化库的型号、固件级别以及其它兼容性方面的信息，请与 Sun 的销售代表或授权服务提供商联系。

本章将介绍以下自动化库：

ADIC/Grau 自动化库

如果您使用的是 ADIC/Grau 自动化库，那么应该按照本节所述的步骤导入和导出卡盒。这些步骤不同于第 13 页的 “在 Sun StorEdge SAM-FS 环境中使用自动化库和手动载入的驱动器”中所述的步骤。

由于从物理上向 ADIC/Grau 自动化库中添加和从中取出卡盒时，使用的是供应商提供的实用程序，因此 Sun StorEdge SAM-FS 接口 （`import`(1M)、`samexport`(1M) 和 SAM-QFS Manager）只能影响库目录。

▼ 导入卡盒

要导入卡盒，请执行以下步骤。

1. 使用 ADIC/Grau 命令将卡盒从物理上移入自动化库。
2. 使用 Sun StorEdge SAM-FS `import`(1M) 命令更新库目录。

   此命令的使用格式如下：

```
import -v volser eq
```

表 3-1　`import`(1M) 命令的变量

变量	含义
volser	要添加的 *volser*（卷序）。`grauaci` 接口在使用新条目更新库目录之前，将验证 ADIC/Grau 自动化库是否具有 *volser* 信息。
eq	所访问的设备在 `mcf` 文件中定义的设备序号。

▼ 导出卡盒

要导出卡盒，请执行以下步骤。

1. **使用 Sun StorEdge SAM-FS `samexport`(1M) 命令从库目录中删除该条目。**
   按以下某一种格式使用此命令：

```
samexport eq:slot
samexport media_type.vsn
```

表 3-2　`samexport`(1M) 命令的变量

变量	含义
eq	所访问的设备在 `mcf` 文件中定义的设备序号。
slot	自动化库中存储端口的编号，与库目录中标识的编号相同。
media_type	介质类型。有关有效介质类型的列表，请参阅 `mcf`(4) 手册页。
vsn	分配给卷的卷序列名。

   导出每一个 VSN 后， `samexport`(1M) 命令将更新库目录，并将每一个 VSN 的库目录条目从库目录移至历史记录。

2. **使用 ADIC/Grau 命令将卡盒从物理上移出自动化库。**

Fujitsu LMF 自动化库

如果您使用的是 Fujitsu LMF 自动化库，那么应该按照本节所述的步骤导入和导出卡盒。这些步骤不同于第 13 页的 “在 Sun StorEdge SAM-FS 环境中使用自动化库和手动载入的驱动器”中所述的步骤。

由于从物理上向 Fujitsu LMF 自动化库中添加或从中取出卡盒时，使用的是供应商提供的实用程序，因此 Sun StorEdge SAM-FS 接口 （`import`(1M)、`samexport`(1M) 和 SAM-QFS Manager）只能影响库目录。

▼ 导入卡盒

要导入卡盒，请执行以下步骤。

1. 使用 Fujitsu 命令将卡盒从物理上移入自动化库。
2. 使用 Sun StorEdge SAM-FS `import`(1M) 命令更新库目录。

   此命令的使用格式如下：

```
import –v volser eq
```

表 3-3　`import`(1M) 命令的变量

变量	含义
volser	要添加的 *volser*（卷序）。`fujitsulmf` 接口在使用新条目更新库目录之前，将验证 LMF 自动化库是否具有 *volser* 信息。
eq	所访问的设备在 `mcf` 文件中定义的设备序号。

▼ 导出卡盒

要导出卡盒，请执行以下步骤。

1. 使用 Sun StorEdge SAM-FS `samexport`(1M) 命令从库目录中删除条目。

   按以下某一种格式使用此命令：

```
samexport eq:slot
samexport media_type.vsn
```

表 3-4　`samexport`(1M) 命令的变量

变量	含义
eq	所访问的设备在 `mcf` 文件中定义的设备序号。
slot	自动化库中存储端口的编号，与库目录中标识的编号相同。
media_type	介质类型。有关有效介质类型的列表，请参阅 `mcf`(4) 手册页。
vsn	分配给卷的卷序列名。

导出每一个 VSN 后， `samexport`(1M) 命令将更新库目录，并将每一个 VSN 的库目录条目从 Sun StorEdge SAM-FS 库目录移至 Sun StorEdge SAM-FS 历史记录。

2. 使用 Fujitsu 命令将卡盒从物理上移出自动化库。

IBM 3584 UltraScalable 磁带库

Sun StorEdge SAM-FS 环境可支持 IBM 3584 UltraScalable 磁带库。以下几节从多方面介绍了此库的操作方法，这些操作不同于第 13 页的 “在 Sun StorEdge SAM-FS 环境中使用自动化库和手动载入的驱动器”中所述的步骤。

导入卡盒

启动 Sun StorEdge SAM-FS 软件时，邮箱中的卡盒不会自动导入。

清洁驱动器

要在 Sun StorEdge SAM-FS 环境中使用此库，应该禁用自动清洁，而启用手动清洁。此过程在 *《IBM 3584 UltraScalable Tape Library Planning and Operator Guide》*（IBM 的出版物 GA32-0408-01）中进行了介绍。 `ibm3584`(7) 手册页中也对此进行了介绍。

分区

此库可以包含多个磁带驱动器。如果您使用了多个驱动器，则可能需要将这一个物理库划分成两个、三个或四个逻辑库。如果您已将此库划分成两个或更多的逻辑库，则在将 IBM 3584 库添加到 Sun StorEdge SAM-FS 环境中之前，应确保这些逻辑库可以正常工作。

从已分区的库中导出卡盒时，只有从中导出卡盒的逻辑库才可以访问该抽屉端口。如果您手动取出卡盒并将其重新插入，则任何逻辑分区均可访问该卡盒。

▼ 取出卡盒

以下步骤说明了在这种情况下所执行的操作：

1. 打开挡门。
2. 取出卡盒。

3. 关闭挡门。
4. 等待挡门锁定，然后取消锁定。
5. 打开挡门。
6. 放回卡盒。
7. 关闭挡门。

有关在 Sun StorEdge SAM-FS 环境中将此库用作逻辑分区库的更多信息，请参阅 IBM 的相关文档或 `ibm3584`(7) 手册页。

IBM 3494 库

Sun StorEdge SAM-FS 环境可支持 IBM 3494 库。以下几节从多方面介绍此库的操作方法，这些操作不同于第 13 页的 “在 Sun StorEdge SAM-FS 环境中使用自动化库和手动载入的驱动器”中所述的步骤。

▼ 导入卡盒

要导入卡盒，请执行以下步骤。

1. 将新的介质放入 I/O 端口中。
2. 关闭挡门。

自动化库将锁定挡门，并介质移至存储区域。一次只能导入 100 个卷。

如果已将库配置为 `access=private`，那么操作到此便可结束。移动介质后这个库会通知后台程序，且介质会被添加到目录中。

3. 使用 `import`(1M) 命令将介质添加到目录中。（可选）

仅在将库配置为 `access=shared` 时才需执行此步骤。

如果已将库配置为 `access=shared`，那么应使用 `import`(1M) 命令将介质添加到目录中。

▼ 导出卡盒

1. 使用 `export`(1M) 命令导出卡盒。

   此命令会将介质移至 I/O 区域，并打开操作员面板上的输出模式指示灯。

2. 从物理上将介质从 I/O 区域中取出。

Sony 8400 PetaSite 直接连接式自动化库

Sony 8400 PetaSite 系列自动化库不同于其它型号的 Sony 产品，因为它配有八端口导入和导出邮箱 （端口编号为 400 至 407）。因此，该系统可以方便快捷地执行导入和导出操作。此自动化库使用条码读取器。

由于邮箱端口可用作存储端口，因此 Sun StorEdge SAM-FS 库目录可以跟踪记录邮箱端口。

注意 – 本节所述的内容仅适用于 Sony 8400 PetaSite 直接连接式自动化库。这些信息既不适用于 Sony B9 和 B35 直接连接式自动化库，也不适用于第 44 页的“Sony 网络连接式自动化库”。

▼ 导入磁带

要导入磁带，请执行以下步骤。

1. 按下自动化库前面板上的打开 / 关闭按钮，打开自动化库的挡门。
2. 将卡盒装入邮箱端口。
3. 按下自动化库前面板上的打开 / 关闭按钮，手动关闭邮箱的挡门。

   自动化库将在关闭挡门后检查邮箱端口，以获取卡盒条码。如果条码有问题，该端口的 `in` （进）和 `out` （出）指示灯均会闪烁。

4. 运行 import(1M) 命令，以使 Sun StorEdge SAM-FS 系统识别导入的卡盒。
此命令的使用格式如下：

```
import eq
```

其中的 *eq*，用于指定所访问的设备在 mcf 文件中定义的设备序号。
另外，还可使用 SAM-QFS Manager 来执行此步骤。

导出磁带

导出磁带卡盒的步骤取决于您是否将邮箱端口用作存储端口。

▼ 在邮箱端口未用作存储端口时导出磁带

在邮箱端口未用作存储端口时，请使用以下步骤导出卡盒。

1. 运行 move(1M) 命令将卡盒移至邮箱端口 （端口编号为 400 至 407）。
此命令的使用格式如下：

```
move source_slot destination_slot eq
```

表 3-5　　move(1M) 命令的变量

变量	含义
source_slot	卡盒当前所在端口的编号。
destination_slot	卡盒将要移至的端口的编号。
eq	所访问的设备在 mcf 文件中定义的设备序号。

2. 按下自动化库前面板上的打开 / 关闭按钮。
此时，挡门会打开。
3. 从邮箱端口中取出卡盒。
4. 按下自动化库前面板上的打开 / 关闭按钮，手动关闭邮箱的挡门。

5. 运行 samexport(1M) 命令，以使 Sun StorEdge SAM-FS 系统识别导出的卡盒。
此命令的使用格式如下：

```
samexport eq
```

其中的 *eq*，用于指定所访问的设备在 mcf 文件中定义的设备序号。
另外，还可使用 SAM-QFS Manager 来执行此步骤。

▼ 在邮箱端口用作存储端口时导出磁带

如果将邮箱端口用作存储端口，并且要导出的卡盒位于其中一个邮箱端口，则请使用以下步骤导出卡盒。

1. 按下自动化库前面板上的打开 / 关闭按钮。
此时，挡门会打开。
2. 从邮箱端口中取出卡盒。
3. 按下自动化库前面板上的打开 / 关闭按钮，手动关闭邮箱的挡门。
4. 运行 samexport(1M) 命令，以使 Sun StorEdge SAM-FS 系统识别导出的卡盒。
此命令的使用格式如下：

```
samexport eq
```

其中的 *eq*，用于指定所访问的设备在 mcf 文件中定义的设备序号。
另外，还可使用 SAM-QFS Manager 来执行此步骤。

▼ 如何将卡盒移至另一个端口

要将卡盒移至另一个端口，请执行以下步骤：

1. 确保来源端口中具有卡盒，而目标端口中没有卡盒。

2. 运行 move(1M) 命令。

此命令的使用格式如下：

```
move eq:source_slot destination_slot
```

表 3-6 move(1M) 命令的变量

变量	含义
eq	所访问的设备在 mcf 文件中定义的设备序号。
source_slot	卡盒当前所在端口的端口编号。
destination_slot	卡盒将要移至的端口编号。

另外，还可使用 SAM-QFS Manager 来执行此步骤。

Sony 网络连接式自动化库

如果您使用的是 Sony 网络连接式自动化库，那么应该按照本节所述的步骤导入和导出卡盒。这些步骤不同于第 13 页的 “在 Sun StorEdge SAM-FS 环境中使用自动化库和手动载入的驱动器”中所述的步骤。

由于从物理上向 Sony 自动化库中添加和从中取出卡盒时，所使用的是供应商提供的实用程序，因此 Sun StorEdge SAM-FS 接口 （import(1M)、 samexport(1M) 和 SAM-QFS Manager）只能影响库目录。

▼ 导入卡盒

要导入卡盒，请执行以下步骤。

1. 使用 Sony 命令将卡盒从物理上移入自动化库。
2. 使用 import(1M) 命令更新库目录。

此命令的使用格式如下：

```
import -v [" ] volser [ " ] eq
```

表 3-7　import(1M) 命令的变量

变量	含义
" "	引号。如果 *volser* 包含空格，则必须包含在引号内。
volser	要添加的 *volser （卷序）*。 PSC API 接口在使用新条目更新库目录之前，将验证 Sony 自动化库是否具有 volser 信息。如果卡盒从物理上并没有位于库中，则会在历史记录目录中添入一个条目。
eq	所访问的库在 mcf 文件中定义的设备序号。

▼ 导出卡盒

要导出卡盒，请执行以下步骤。

1. **使用 samexport(1M) 命令从库目录中删除条目。**

   按以下某一种格式使用此命令：

```
samexport eq:slot
samexport media_type.vsn
```

表 3-8　samexport(1M) 命令的变量

变量	含义
eq	所访问的设备在 mcf 文件中定义的设备序号。
slot	自动化库中存储端口的编号，与库目录中标识的编号相同。
media_type	介质类型。有关有效介质类型的列表，请参阅 mcf(4) 手册页。
vsn	分配给卷的卷序列名。

导出每一个 VSN 后， samexport(1M) 命令将更新库目录，并将每一个 VSN 的库目录条目从库目录移至历史记录。

2. **使用 Sony 命令将卡盒从物理上移出自动化库。**

StorageTek ACSLS 连接式自动化库

如果您使用的是 StorageTek ACSLS 连接式自动化库，那么应该按照本节所述的步骤导入和导出卡盒。这些步骤不同于第 13 页的 “在 Sun StorEdge SAM-FS 环境中使用自动化库和手动载入的驱动器”中所述的步骤。

邮箱 是自动化库中的一个区域，它用于向自动化库中添加和从中取出卡盒。某些 StorageTek 自动化库一次只能导入和导出一个卡盒。StorageTek 9714 和 StorageTek 9710 就属于受 Sun StorEdge SAM-FS 环境支持的带邮箱的 StorageTek 自动化库。StorageTek 9730 使用邮箱端口。在 StorageTek 文档中，通常将邮箱和邮箱端口称为 *CAP*。

在向 ACSLS 连接式自动化库中导入和从中导出卡盒时，请记住以下几点：

- 导入卡盒时， Sun StorEdge SAM-FS 命令仅影响库目录。 `import`(1M) 命令并不会从物理上将卡盒插入自动化库中。您必须使用 ACSLS 命令才可以从物理上导入卡盒。
- 导出卡盒时，除非在 `samexport`(1M) 命令中使用 `-f` 选项，否则 Sun StorEdge SAM-FS 命令仅影响库目录。使用 `-f` 选项后， Sun StorEdge SAM-FS 系统会将卷放入卡盒访问端口 (CAP) 中，并相应地更新目录。如果不使用 `-f` 选项，则仅更新目录而不会将卷放入 CAP 中，因此您仍必须使用 ACSLS 命令从物理上导出卡盒。

按照协议，应该由您来负责维护 ACSLS 清单和 Sun StorEdge SAM-FS 目录。

另外，还可使用 `samu`(1M) 或 SAM-QFS Manager 来执行导入和导出过程。

▼ 导入磁带

● **要导入磁带卡盒，请使用 `import`(1M) 命令。**

此命令的使用格式如下：

```
import -v vsn eq
```

表 3-9　`import`(1M) 命令的变量

变量	含义
vsn	分配给卷的卷序列名。
eq	所访问的设备在 `mcf` 文件中定义的设备序号。

import(1M) 命令将在库目录中添入一个新的 VSN。如果历史记录中包含此 VSN，则 Sun StorEdge SAM-FS 软件会将 VSN 信息从历史记录移至库目录。

▼ 使用邮箱导出磁带

您既可以按端口导出磁带卡盒，也可以按 VSN 导出磁带卡盒。

- **要导出磁带卡盒，请使用 samexport(1M) 命令。**

  按以下某一种格式使用此命令：

```
samexport [-f] eq:slot
samexport [-f] media_type.vsn
```

表 3-10　samexport(1M) 命令的变量

变量	含义
-f	命令 Sun StorEdge SAM-FS 系统将卷放入卡盒访问端口 (CAP) 中，并相应地更新目录。
eq	所访问的设备在 mcf 文件中定义的设备序号。
slot	自动化库中存储端口的编号，与库目录中标识的编号相同。
media_type	介质类型。有关有效介质类型的列表，请参阅 mcf(4) 手册页。
vsn	分配给卷的卷序列名。

导出每一个 VSN 后， samexport(1M) 命令将更新库目录，并将每一个 VSN 的库目录条目从库目录移至历史记录。

第四章

存档

存档 是指将文件从 Sun StorEdge SAM-FS 文件系统复制到可移动介质卡盒上的卷或另一个文件系统中磁盘分区的过程。本章中，术语 *存档介质* 是指向其中写入存档卷的各种卡盒或磁盘位片。Sun StorEdge SAM-FS 的存档功能可完成许多操作，例如可指定哪些文件需要立即存档，可指定哪些文件永不需要存档，还可执行其它任务。

本章介绍存档程序的操作原理、制订站点存档策略的通用原则，以及如何通过创建 `archiver.cmd` 文件来实施这些存档策略。

它包括下列主题：

存档程序 — 操作原理

存档程序自动将 Sun StorEdge SAM-FS 文件写入存档介质中。存档和登台文件并不需要操作人员的参与。文件存档至存档介质上的卷，每个卷由称为 *卷序列名* (VSN, Volume Serial Name) 的唯一标识来识别。存档介质中可以包含一个或多个卷。识别单个卷时，必须指定介质类型和 VSN。

存档程序在 Sun StorEdge SAM-FS 文件系统安装时自动启动。可以通过在下面的文件中插入存档指令，来为站点自定义存档程序的操作：

```
/etc/opt/SUNWsamfs/archiver.cmd
```

执行存档时，并不一定需要有 `archiver.cmd` 文件。当没有这个文件时，存档程序使用下列默认设置：

- 将所有文件存档至可用卷。
- 所有文件的*存档时限*为 4 分钟。存档时限是指从最后一次修改文件到开始存档文件所经历的时间。
- *存档时间间隔*为 10 分钟。存档时间间隔是指两次完整的存档过程之间相隔的时间。

以下几节介绍存档组的概念，并说明在存档过程中所执行的操作。

存档组

*存档组*表示一组要存档的文件。用户可以在任意多个文件系统上定义存档组。同一个存档组中的文件在大小、所有权、组和目录位置等方面遵守相同的标准。存档组控制存档副本的目的地、存档副本的保留时间以及数据存档之前的等待时间。存档组中的所有文件均被复制到与该存档组关联的卷。文件系统中的一个文件能且只能属于一个存档组。

文件在创建或修改之后，存档程序会将其复制到存档介质中。存档文件与标准 UNIX `tar`(1) 格式兼容。这确保了 Sun Solaris 操作系统 (OS) 与其它 UNIX 系统之间的数据兼容性。此格式包括文件访问数据 （即索引节点信息）和文件的路径。由于采用了 `tar`(1) 格式，这样在 Sun StorEdge SAM-FS 环境中的所有数据都丢失时，就可使用标准的 UNIX 工具和命令来恢复文件。存档过程还会复制执行 Sun StorEdge SAM-FS 文件系统操作所必需的数据。此类数据包括目录、符号链接、分段文件的索引和存档介质信息。

在本节的后半部分，术语*文件*既指文件数据，也指元数据。仅在需要加以区分时，才会使用术语*文件数据*和*元数据*。而术语*文件系统*是指已安装的 Sun StorEdge SAM-FS 文件系统。

通常，管理员可随意地为存档组命名，但以下情况除外：

- 存档组有两个保留名称：`no_archive` 和 `allsets`。

  `no_archive` 存档组是系统默认定义的存档组。选入此存档组的文件永远不会被存档。临时目录 （例如 `/sam1/tmp`）中的文件可能会包括在 `no_archive` 存档组中。

  `allsets` 存档组用于定义适用于所有存档组的参数。

- 每个 Sun StorEdge SAM-FS 文件系统的存档组名称都会被保留，以便控制结构信息。 Sun StorEdge SAM-FS 文件系统为每个文件系统都提供了一个默认存档组。对于每个文件系统，存档程序不仅为其存档元数据，而且还存档文件数据。文件系统存档组包括目录和链接信息以及不属于其它存档组的任何文件。默认存档组的名称与其关联的文件系统的名称相同，并且不能更改。例如，如果一个文件系统的名称为 samfs1，则其存档组的名称应为 samfs1。
- 存档组名最长不得超过 29 个字符。所用的字符仅限于 26 个大小写字母、数字 0 至 9 以及下划线字符 (_)。

存档操作

默认情况下，存档程序将为每一个存档组创建一份副本，但您可以要求为每一个存档组创建多达四份副本。存档组和副本份数意味着占用一系列卷。存档副本在其它卷中提供文件的复件。

为确保完整无缺地存档文件，存档程序会在文件修改之后等待一段指定的时间，然后再对其进行存档。正如前文所述，这段时间称为*存档时限*。

文件中的数据须经更改后，该文件才被视为存档或重新存档的对象。如果只是访问文件，则不会对其存档。例如，对某个文件运行 touch(1) 或 mv(1) 命令，并不会导致系统对该文件进行存档或重新存档。运行 mv(1) 命令只是改变文件的名称，而并没有更改文件数据。在故障恢复时，如果通过 tar(1) 文件进行恢复，则运行此命令可能会产生其它后果。有关故障恢复的详细信息，请参阅 *《Sun QFS、 Sun SAM-FS 和 Sun SAM-QFS 故障恢复指南》*。

存档程序依据文件的存档时限来选择要存档的文件。用户可以为每一个存档副本定义存档时限。

用户可以使用 touch(1) 命令将其文件的默认时间参考，更改成过去较早的时间或将来较远的时间。但是，这可能会导致意外的存档结果。为了避免此类问题，存档程序会调节时间参考以使它们始终处于下面所示的范围：

creation_time < *time_ref* < *time_now*

以下几节介绍存档程序执行的步骤，包括从最初的文件扫描进程到文件复制进程。

步骤 1：识别要存档的文件

每一个已安装的文件系统均有单独的 sam-arfind 进程。sam-arfind 进程将监控每一个文件系统，以确定哪些文件需要存档。当某文件的更改会影响它自身的存档状态时，文件系统都会向其 sam-arfind 进程发出通知。这类更改的示例有：对文件进行修改、重新存档、取消存档以及重命名。sam-arfind 进程在接到通知后会检查该文件，以确定是否需要对它执行存档操作。

sam-arfind 进程根据文件的属性说明，来确定文件所属的存档组。用于确定文件所属存档组的特征如下：

- 文件名中的目录路径部分，以及采用了正则表达式的完整文件名 （可选）
- 文件所有者的用户名
- 文件所有者的组名
- 最小文件大小
- 最大文件大小

如果文件的一份或多份副本到达或超出存档时限，sam-arfind 会将文件添加到存档组的一个或多个存档请求中。*存档请求*是属于同一个存档组的所有文件的集合。对于要重新存档的文件，则采用单独的存档请求。这样可以分别控制尚未存档文件以及重新存档文件的时间安排。存档请求是一个文件，它位于下面的目录中：

```
/var/opt/SUNWsamfs/archiver/file_sys/ArchReq
```

这个目录中的文件是二进制文件，可以使用 showqueue(1M) 命令显示。

存档请求有时也称为 *ArchReq*。

如果文件的一份或多份副本未达到存档时限，该文件所在的目录以及将到达存档时限的时间会被添加到扫描列表中。一旦到达扫描列表中列出的时间，将对目录进行扫描。并将到达存档时限的文件添加到存档请求中。

如果文件处于离线状态，sam-arfind 进程会选择将用作存档副本来源的卷。如果文件副本需要重新存档，sam-arfind 进程将选择包含这个需要重新存档的存档副本的卷。

如果文件已被分成数段，此进程将仅选择已发生更改的文件段进行存档。分段文件的索引不含用户数据，因此这些分段文件将被视为文件系统存档组的成员而单独进行存档。

存档优先级是根据文件属性特征以及与存档组关联的文件属性乘数计算出来的。事实上，其计算公式如下所示：

archive_priority = (*file_property_value* * *property_multiplier*) 之和

大多数 *file_property_value* 数值为 1 或 0，相当于属性是 TRUE 还是 FALSE。例如，如果正在创建第 1 个存档副本，则第 1 个属性副本的值为 1。而第 2 个副本、第 3 个副本和第 4 个副本的属性值为 0。

存档时限和文件大小等其它文件属性的值可以是 0 或 1 之外的其它数值。

property_multiplier 的值由存档组的 -priority 参数决定。由于可为文件的各个方面 （例如存档时限或大小）设定值，因此您可以改变站点的存档请求的优先级。有关 -priority 参数的详细信息，请参阅 archiver.cmd(4) 手册页。

archive_priority 和属性乘数是浮点数值。所有属性乘数的默认值均为 0.0。此时，此存档请求被设置为存档请求中的最高文件优先级。

确定文件是否需要存档的方法有两种：连续存档和扫描。连续存档即为，存档程序会对文件系统进行处理，以确定哪些文件需要存档。而扫描方法则是，存档程序定期细查文件系统，并选择需要存档的文件。以下几节将介绍这两种方法。

连续存档

连续存档是默认的存档方法 (examine=noscan)。在连续存档方法中，可以通过 -startage、 -startcount 以及 -startsize 参数来指定存档组开始存档的条件。这些条件可以是优化存档时间，也可以是优化所完成的存档工作量。

- 示例 1。 如果需要花费一个小时来创建应该同时存档的文件，那么可以将 -startage 参数设置为 1 小时 (-startage 1h)，以确保所有文件在安排存档请求前都已创建完毕。
- 示例 2。 可以将 -startsize 指定为 150 GB (-startsize 150g)，来命令存档程序在需要存档的数据达到 150 GB 时，才进行存档。
- 示例 3。 如果知道要生成 3000 个需要存档的文件，那么可以通过指定 -startcount 3000，来确保同时对这些文件进行存档。

在满足任何一个启动条件后，sam-arfind 进程就会发送存档请求至存档后台程序 sam-archiverd，以安排将文件复制到存档介质。

扫描存档

如果不使用连续存档方法，那么可以指定 examine=scan 来指示 sam-arfind 进行扫描，以检查需要存档的文件。需要存档的文件会被放入存档请求中。sam-arfind 进程将定期扫描每一个文件系统，以确定哪些文件需要存档。sam-arfind 进程执行的第一个扫描过程是目录扫描。在此扫描期间，sam-arfind 按降序方式逐层搜寻目录树。它将检查每一个文件，如果文件不需要存档，则为其设置 archdone（已存档）文件状态标记。接下来，对 .inodes 文件进行扫描。此时，只检查那些没有 archdone 标记的索引节点。

完成文件系统扫描之后，sam-arfind 进程会将每一个存档请求发送至存档程序的后台程序 sam-archiverd，以安排将文件复制到存档介质。sam-arfind 进程随后进入休眠状态，休眠的时间为 interval=*time* 指令所指定的时间。在时间间隔结束时， sam-arfind 进程将重新开始扫描。

步骤 2：编辑存档请求

sam-archiverd 后台程序收到存档请求后，将对其进行*编辑*。下面介绍这个编辑过程。

可能无法一次对存档请求中的所有文件进行存档。这取决于存档介质的容量或存档程序命令文件中指定的控制条件。*编辑* 是指从存档请求中选择一次要对哪些文件进行存档的过程。完成存档请求的存档复制操作后，如果仍有需要存档的文件，则会重新编辑存档请求。

sam-archiverd 后台程序会根据某些默认标准和站点特定的标准，来排列存档请求中的文件。默认操作是根据文件系统扫描期间文件的发现顺序，依次将存档请求中的所有文件存档到同一个存档卷中。您可以通过指定站点的特定标准，来控制按什么顺序对文件进行存档，以及如何将它们分布到不同的卷中。 这些标准称为*存档组参数*，其检验顺序依次为：-reserve、 -join、 -sort、 -rsort （执行逆向排序）以及 -drives。有关这些参数的详细信息，请参阅 archiver.cmd(4) 手册页。

如果存档请求属于已指定 -reserve *owner* 的存档组，则 sam-archiverd 后台程序将依据文件的目录路径、用户名或组名来排列存档请求中的文件。此操作由存档组的 -reserve 参数控制。它首先选择属于第一个 *owner* 的文件进行存档。剩余文件将在以后进行存档。

如果存档请求属于已指定 -join *method* 的存档组，则 sam-archiverd 后台程序将根据指定的 -join *method* 对文件进行分组。如果还指定了 -sort 或 -rsort *method* ，那么 sam-archiverd 后台程序将根据 -sort 或 -rsort *method* 排列每个组中的文件。 此时，存档请求既进行了组合，也进行了排序。

对于以后的编辑和安排进程，每一个组内的所有组合文件均被视为单个文件。

如果存档请求属于已指定 -sort 或 -rsort *method* 的存档组，则 sam-archiverd 后台程序将根据 -sort 或 -rsort 参数指定的排序方法排列文件。 sam-archiverd 后台程序通常会根据排序方法、存档时限、大小或目录位置将文件组合在一起，具体视排序方法而定。默认情况下，存档请求不会进行排序，因此存档程序将根据在文件系统扫描期间发现文件的顺序对文件进行存档。

sam-archiverd 后台程序可以确定文件是处于在线状态还是离线状态。如果存档请求中既包含在线文件，也包含离线文件，则会首先选择在线文件进行存档。

如果未要求存档请求按一定的排序方法进行组合或排序，则离线文件会按存档副本所在的卷来排列。这可确保同一卷中每一个存档组内的所有文件可以按它们在介质上的排列顺序同时登台。在为离线文件创建多份存档副本期间，离线文件不会被释放，直到创建完所有要求的副本。与第一个文件处于同一卷中且要登台的所有文件都将被选作首先进行存档的文件。

请注意，在对离线文件进行存档时，使用 -join、 -sort 或 -rsort 参数可能会对性能造成负面影响。这是因为要存档的文件的顺序可能与离线文件所需的存档卷顺序不符。因此，建议您仅在创建第一份存档副本时使用 -join、 -sort 或 -rsort 参数。开始创建其它副本时，如果存档介质有足够的空间，则其它副本会尽可能地采用第一份副本的存档顺序进行存档。

存档请求将输入至 sam-archiverd 后台程序的安排队列中。

步骤 3：安排存档请求

sam-archiverd 后台程序中的安排程序将在出现下列情况时立即执行：

- 存档请求已输入至安排队列中。
- 已完成某个存档请求的存档。
- 从目录服务器收到介质状态发生变化的消息。
- 收到更改存档程序状态的消息。

安排队列中的存档请求按优先级排列。安排程序每次执行时，均会检查所有存档请求，以确定是否可以将它们分配至 sam-arcopy 进程，从而将文件复制到存档介质中。

此时，必须存在用于创建文件副本的驱动器。必须存在可供存档组使用的卷，并且它们有足够的空间来容纳存档请求中的文件。

驱动器

如果存档组指定了 -drives 参数，则 sam-archiverd 后台程序会将存档请求中选定的文件分配至多个驱动器中。如果此时可用的驱动器数量少于 -drives 参数指定的数量，则使用实际可用的数量。

如果存档请求中的文件大小总量小于 -drivemin 值，则只使用一个驱动器。-drivemin 值可以是 -drivemin 参数指定的值，也可以是 archmax 值。

archmax 值是由 -archmax 参数指定的值或为此介质定义的值。有关 -archmax 参数和 archmax= 指令的详细信息，请参阅 archiver.cmd(4) 手册页。

如果存档请求中的文件大小总量大于 -drivemin 值，则会执行以下计算：*drive_count* = *total_size* / *drivemin*。如果 *drive_count* 小于 -drives 参数所指定的驱动器数，则使用 *drive_count* 所指定的驱动器数。

不同的文件存档方式，花在驱动器上的时间各不相同。可以使用 -drivemax 参数以获得更好的驱动器利用率。-drivemax 参数要求您在重新安排驱动器接受更多数据前，指定将写入该驱动器的最大字节数。

卷

系统必须存在一个或多个具有足够空间的卷，来容纳存档请求中的全部文件或至少一部分文件。如果存档组最近使用过的卷上有足够的空间，存档程序将使用这个卷。另外，该卷不应是存档程序正在使用的卷。

如果可用于存档组的卷正在使用中，则存档程序会选择其它卷。但只有在未指定 `-fillvsns` 参数时，此原则才适用。如果指定了该参数，存档请求不能另行安排。

如果存档请求太大，无法装入一个卷中，则系统会选择只将这个卷所能容纳的文件存档于其中。如果存档请求包含的文件太大，无法装入一个卷中，并且未为存档请求选择卷溢出功能，则这些文件无法存档。此时，系统会将一则说明此情况的消息发送至日志中。

您可以使用 `-ovflmin` 参数为存档组指定卷溢出功能，或使用 `ovflmin=` 指令为介质指定卷溢出功能。有关 `-ovflmin` 参数和 `ovflmin=` 指令的详细信息，请参阅 `archiver.cmd`(4) 手册页。此参数 `ovflmin` 用于确定文件溢出介质的最小大小。为存档组指定的 `ovflmin` 优先于为介质定义的 `ovflmin`。如果文件的大小小于 `ovflmin`，则文件无法存档。此时，系统会将一则说明此情况的消息发送至日志中。

如果文件的大小大于 `ovflmin`，则会根据需要分配其它卷。系统将按容量逐渐减少的顺序选择其它卷，以尽可能地减少文件所占用的卷数量。

如果没有可用于存档请求的卷，存档请求会等待。

在确定某个存档请求的安排优先级时，除依据步骤 1 中得出的存档优先级之外，系统还将参照某些其它属性，例如文件是处于在线状态还是处于离线状态。有关自定义属性乘数的详细信息，请参阅 `archiver.cmd`(4) 手册页中所述的 `-priority` 参数。

对于每一个存档请求， `sam-archiverd` 后台程序均会通过计算存档优先级和各种与系统资源属性相关联的乘数之和，来确定其安排优先级。这些资源属性与以下各项相关：存档请求排队的时间 （秒数）；存档进程中使用的第一个卷是否已载入驱动器；以及其它方面。

使用经调整的优先级， `sam-archiverd` 后台程序指定每一个可进行复制的存档请求。

步骤 4：对存档请求中的文件进行存档

当存档请求已经准备就绪，可以进行存档时， sam-archiverd 后台程序将检查每一个存档请求，标记存档文件 (tarball) 的界限，以使每一个存档文件的大小不超过 -archmax *target_size* 参数指定的值。如果单个文件大于 *target_size*，它将成为存档文件中的唯一文件。

对于每一个存档请求和要使用的每一个驱动器， sam-archiverd 后台程序均会将存档请求分配至 sam-arcopy 进程，以便将文件复制到存档介质。如果单个文件大于 *target_size*，它将成为存档文件中的唯一文件。存档信息会输入至索引节点。

如果已启用存档日志功能，则会创建存档日志条目。

如果文件已登台，系统会释放其磁盘空间。此进程会持续运行，直到列表中的所有文件存档完毕。

多种错误以及文件状态改变都会导致文件复制失败。这其中包括读取磁盘高速缓存或向卷写入数据时发生的错误。另外，文件状态的改变也会导致文件复制失败，其中包括在选择文件后对文件进行了修改，打开文件并写入了数据，以及文件被删除等。

sam-arcopy 进程退出后，sam-archiverd 后台程序将检查存档请求。如果有文件尚未存档，则会重新编辑存档请求。

默认输出范例

代码示例 4-1 显示了运行 archiver(1M) -l 命令的输出范例。

代码示例 4-1 archiver(1M) -l 命令的输出

```
# archiver

Archive media:
default:mo
media:mo archmax:5000000
media:lt archmax:50000000
Archive devices:
device:mo20 drives_available:1 archive_drives:1
device:lt30 drives_available:1 archive_drives:1
Archive file selections:
Filesystem samfs1:
samfs1  Metadata
    copy:1  arch_age:240
big  path:. minsize:512000
    copy:1  arch_age:240
```

代码示例 4-1 archiver(1M) -l 命令的输出 *（接上页）*

```
all  path:
     copy:1  arch_age:30
Archive sets:
all
     copy:1 media:mo
big
     copy:1 media:lt
samfs1
     copy:1 media:mo
```

存档程序的后台程序

sam-archiverd 后台程序用于安排存档活动。sam-arfind 进程用于将需要存档的文件分配至存档组。sam-arcopy 进程用于将需要存档的文件复制到选定的卷。

一旦 Sun StorEdge SAM-FS 开始活动， sam-fsd 就会启动 sam-archiverd 后台程序。sam-archiver 后台程序将执行 archiver(1M) 命令，以读取 archiver.cmd 文件并建立用于控制存档操作的表。它将为每一个已安装的文件系统启动 sam-arfind 进程；同样，如果未安装文件系统，则会停止相关的 sam-arfind 进程。然后， sam-archiverd 进程将监控 sam-arfind 的运行状况，并处理来自操作员或其它进程的信号。

存档日志文件和事件日志

sam-arfind 和 sam-arcopy 进程可以生成日志文件，该文件中包含每一个已存档文件或自动取消存档的文件的有关信息。日志文件连续地记录存档操作。您可以使用日志文件查找文件的早期副本，这些副本是传统备份的产物。

默认情况下，系统不会生成此文件。您可以在 archiver.cmd 文件中插入 logfile= 指令，以指明创建日志文件，并指定其名称。此文件的名称由您指定。有关日志文件的详细信息，请参阅本章第 65 页的 “archiver.cmd 指令”和 archiver.cmd(4) 手册页。

存档程序使用 syslog 工具和 archiver.sh，在日志文件中记录警告及参考性消息。

代码示例 4-2 下面是一些摘自存档程序日志文件的示例行，这些行包括了每个字段的定义。

代码示例 4-2 存档程序日志文件的内容

```
A 2001/03/23 18:42:06 mo 0004A arset0.1 9a089.1329 samfs1
118.51162514 t0/fdn f 0 56
A 2001/03/23 18:42:10 mo 0004A arset0.1 9aac2.1 samfs1 189.53
1515016 t0/fae f 0 56
A 2001/03/23 18:42:10 mo 0004A arset0.1 9aac2.b92 samfs1 125.53
867101 t0/fai f 0 56
A 2001/03/23 19:13:09 lt SLOT22 arset0.2 798.1 samfs1 71531.14
1841087 t0/fhh f 0 51
A 2001/03/23 19:13:10 lt SLOT22 arset0.2 798.e0e samfs1 71532.12
543390 t0/fhg f 0 51
A 2003/10/23 13:30:24 dk DISK01/d8/d16/f216 arset4.1 810d8.1 qfs2
119571.301 1136048 t1/fileem f 0 0
A 2003/10/23 13:30:25 dk DISK01/d8/d16/f216 arset4.1 810d8.8ad
qfs2 119573.295 1849474 t1/fileud f 0 0
A 2003/10/23 13:30:25 dk DISK01/d8/d16/f216 arset4.1 810d8.16cb
qfs2 119576.301 644930 t1/fileen f 0 0
A 2003/10/23 13:30:25 dk DISK01/d8/d16/f216 arset4.1 810d8.1bb8
qfs2 119577.301 1322899 t1/fileeo f 0 0
```

表 4-1 按从左至右的顺序列出了上面示例中各字段的内容。

表 4-1 存档程序日志文件的字段

字段	内容
1	存档活动，如下所示： • A 表示已存档。 • R 表示已重新存档。 • U 表示已取消存档。
2	存档操作发生的日期，格式为 *yyyy*/*mm*/*dd*。
3	存档活动发生的时间，格式为 *hh*:*mm*:*ss*。
4	存档介质类型。有关介质类型的详细信息，请参阅 mcf(4) 手册页。
5	VSN。对于可移动介质卡盒，这是卷序列名。对于磁盘存档，这是磁盘卷名以及 tar(1) 存档文件的路径。
6	存档组和副本份数。
7	存档文件 （tar(1) 文件）在介质上的起始物理位置和存档文件中的文件偏移量（采用十六进制表示）。
8	文件系统名。

表 4-1　存档程序日志文件的字段　*（接上页）*

字段	内容
9	索引节点编号和世代编号。世代编号是在索引编号被重新使用后生成的一个附加编号，它与索引编号一起用来标识使用的唯一性。
10	在文件仅写入一个卷时，表示文件大小。在文件写入多个卷时，表示部分文件的大小。
11	相对于文件系统安装点的文件路径以及名称。
12	文件类型，如下所示： • `d` 表示目录。 • `f` 表示普通文件。 • `l` 表示符号链接。 • `R` 表示可移动的介质文件。 • `I` 表示段索引。 • `S` 表示数据段。
13	溢出文件的一部分或段。如果这个文件是溢出文件，该值不为零 而对于其他所有文件类型，该值均为零。
14	文件存档至的驱动器的设备序号。

`archiver.cmd` 文件

`archiver.cmd` 文件用于控制存档程序的行为方式。默认情况下，只要一启动 `sam-fsd` 或安装 Sun StorEdge SAM-FS 文件系统，存档程序就开始运行。存档程序默认情况下执行下列操作：

- 将所有文件存档至所有可用的卷。
- 所有文件的存档时限为 4 分钟。
- 存档时间间隔为 10 分钟。

您很可能希望对存档程序的操作进行自定义，以满足站点的特殊存档要求。这些操作由存档程序命令文件 (`archiver.cmd`) 中的指令控制。

▼ 创建或修改 archiver.cmd 文件并应用更改

1. **确定是否需要编辑 archiver.cmd 文件，或是否需要编辑临时的 archiver.cmd 文件。（可选）**

   如果您有 /etc/opt/SUNWsamfs/archiver.cmd 文件，且系统已在对文件进行存档，则执行此步骤。可考虑将 archiver.cmd 文件复制到一个临时位置，以利于对它进行编辑并在正式使用前进行测试。

2. **使用 vi(1) 或其它编辑器来编辑 archiver.cmd 文件或其临时文件。**

   为了控制您的站点的存档操作，可根据您的需要添加指令。有关可在这个文件中添加哪些指令的信息，请参阅第 65 页的 “archiver.cmd 指令”和第 101 页的 “磁盘存档”。

3. **保存并关闭 archiver.cmd 文件或其临时文件。**

4. **使用 archiver(1M) -lv 命令验证文件的正确性。**

   无论您何时更改 archiver.cmd 文件，均应使用 archiver(1M) 命令检查其中是否存在语法错误。使用如下所示的 archiver(1M) 命令，在当前 Sun StorEdge SAM-FS 系统中检验 archiver.cmd 文件：

   ```
   # archiver -lv
   ```

   上面的命令将列出所有选项，并将 archiver.cmd 文件、卷、文件系统内容和错误的列表写入标准输出文件 (stdout)。出现错误时，存档程序将停止运行。

   默认情况下， archiver(1M) 命令会检验 /etc/opt/SUNWsamfs/archiver.cmd 文件中的错误。如果您修改的是 archiver.cmd 文件的临时文件，那么可在 archiver(1M) 命令中使用 -c 选项并提供这个临时文件的名称，以在正式使用之前对其进行检验。

5. **重复步骤 2、步骤 3 以及步骤 4，直到文件没有错误。**

   在执行下一步骤前，必须更正所有错误。如果存档程序发现 `archiver.cmd` 文件中有错误，那么它不会对任何文件进行存档。

6. **将这个临时文件移至** `/etc/opt/SUNWsamfs/archiver.cmd`**。（可选）**

   仅当您使用了临时文件时，才需要执行此步骤。

7. **保存并关闭** `archiver.cmd` **文件。**

8. **使用** `samd`(1M) `config` **命令应用文件更改并重新启动系统。**

```
# samd config
```

`archiver.cmd` 文件

`archiver.cmd` 文件由以下类型的指令组成：

- 全局指令
- 存档组分配指令
- 存档组指令
- VSN 池指令
- VSN 关联指令

这些指令由从 `archiver.cmd` 文件读取的文本行组成。每一行指令包含一个或多个字段，它们由空格或制表符隔开。井字符 (#) 后面的任何文本均被视为注释，并且不会受到检查。在行的末尾添加一个反斜杠 (\)，可使该行续接至下一行。

archiver.cmd 文件中的某些指令会要求您指定时间单位或字节单位。要指定这些单位，请将第 63 页的表 4-2 “archiver.cmd 文件指令单位”中用于代表单位的字母作为数字的后缀。

表 4-2 archiver.cmd 文件指令单位

单位后缀	含义
时间后缀：	
s	秒。
m	分钟，60 秒。
h	小时，3,600 秒。
d	天，86,400 秒。
w	周，604,800 秒。
y	年，31,536,000 秒。
大小后缀：	
b	字节。
k	千字节 (KB)，2**10 或 1,024 字节。
M	兆字节 (MB)，2**20 或 1,048,576 字节。
G	千兆字节 (GB)，2**30 或 1,073,741,824 字节。
T	兆兆字节 (TB)，2**40 或 1,099,511,627,776 字节。
P	千兆兆字节 (PT)，2**50，或 1,125,899,906,842,624 字节。
E	兆兆兆字节 (ET)，2**60，或 1,152,921,504,606,846,976 字节。

archiver.cmd 文件示例

代码示例 4-3 显示了 archiver.cmd 文件示例。右侧的注释指出了各种指令类型。

代码示例 4-3 archiver.cmd 文件示例

```
interval = 30m                          # 全局指令
logfile = /var/opt/SUNWsamfs/archiver/archiver.log

fs = samfs1                             # 存档组分配
no_archive tmp
work work
    1 1h
    2 3h
images images -minsize 100m
    1 1d
    2 1w
samfs1_all  .
    1 1h
    2 1h

fs = samfs2                             # 存档组分配
no_archive tmp
system . -group sysadmin
    1 30m
    2 1h
samfs2_all  .
    1 10m
    2 2h

params                                  # 存档组指令
allsets -drives 2
images.1 -join path -sort size
endparams
vsns                                    # VSN 关联
samfs1.1   mo      optic-2A
samfs1.2   lt      TAPE01
work.1     mo      optic-[3-9][A-Z]
work.2     lt      .*
images.1   lt      TAPE2[0-9]
images.2   lt      TAPE3[0-9]
samfs1_all.1       mo.*
samfs1_all.2       lt.*
samfs2.1   mo      optic-2A
samfs2.2   lt      TAPE01
system.1   mo      optic08a optic08b
system.2   lt      ^TAPE4[0-1]
samfs2_all.1       mo.*
samfs2_all.2       lt.*
endvsns
```

archiver.cmd 指令

以下几节对 archiver.cmd 指令进行介绍。这些指令是：

全局存档指令

全局指令控制存档程序的整体操作。archiver.cmd 文件中的全局指令可以通过第二个字段中的等号 (=) 或缺少其它字段来识别。这些指令可以使您根据站点的配置来优化存档程序的操作。

在 archiver.cmd 文件中，全局指令必须位于所有 fs= 指令前面。fs= 指令是那些专用于特定文件系统的指令。如果存档程序检测到全局指令位于 fs= 指令后面，则会发出一则消息。

archivemeta 指令：控制是否对元数据进行存档

archivemeta 指令控制是否对文件系统元数据进行存档。如果您文件系统中的文件经常移动，且目录结构经常发生更改，那么应该对元数据进行存档。但是，如果目录结构非常稳定，那么可以不对元数据进行存档，或在载入和卸载卡盒以对元数据进行存档时，减少可移动介质驱动器所执行的操作。默认情况下，会对元数据进行存档。

此指令的格式如下：

```
archivemeta = state
```

其中的 *state*，用于指定状态为 on （开）还是 off （关）。默认设置是 on。

元数据的存档方式不尽相同，这取决于您所使用的超级块是第 1 版还是第 2 版，具体如下：

- 如果使用的是第 1 版的文件系统，存档程序会将目录、可移动的介质文件、段索引节点以及符号链接存档为元数据。
- 如果使用的是第 2 版的文件系统，可移动的介质文件和符号链接会存储在索引节点中，而不是存储在数据块中。存档程序不会对它们进行存档。存档程序只将目录和段索引节点存档为元数据。符号链接则存档为数据。

archmax 指令：控制存档文件的大小

archmax 指令用于指定存档文件的最大大小。将用户文件组合在一起，形成存档文件。达到 *target_size* 值后，将不能再向存档文件中添加用户文件。大型用户文件将写入单个存档文件中。

要更改默认值，请使用下面的指令：

```
archmax=media target_size
```

表 4-3　archmax 指令的变量

变量	含义
media	介质类型。有关有效介质类型的列表，请参阅 mcf(4) 手册页。
target_size	指定存档文件的最大大小。存档文件的最大大小与介质有关。默认情况下，写入光盘的存档文件不得超过 5 MB。对于磁带，存档文件的最大默认大小为 512 MB。

将存档文件的大小设置为较大的值或较小的值都各有优缺点。例如，在使用磁带进行存档时，将 archmax 设置成较大的值，可以减少磁带驱动器停止和启动的次数。但是，在写入较大的存档文件时，有可能提前到达磁带末尾，因而浪费了大量磁带空间。一般而言， archmax 的设置值不应超过介质容量的百分之五。例如，可以使用如下所示的 archmax 指令来设置 20 GB 的磁带：

```
archmax=sg 1G
```

此外，您还可为单个存档组设置 archmax 指令。

bufsize 指令：设置存档程序缓冲区大小

默认情况下，系统使用内存缓冲区将需要存档的文件复制到存档介质。您可以使用 bufsize 指令来设置非默认的缓冲区大小和锁定缓冲区 （可选）。这些操作可以改善系统的性能。您可以尝试不同的 *buffer_size* 值以确定最适合的缓冲区大小。

此指令的格式如下：

```
bufsize=media  buffer_size  [ lock ]
```

表 4-4 bufsize 指令的变量

变量	含义
media	介质类型。有关有效介质类型的列表，请参阅 mcf(4) 手册页。
buffer_size	指定一个 2 至 32 的值。默认值为 4。这个值乘以该介质类型的 *dev*_blksize 值，计算结果即为所使用的缓冲区大小。您可以在 defaults.conf 文件中指定 *dev*_blksize 的值。有关此文件的详细信息，请参阅 defaults.conf(4) 手册页。
lock	lock 变量指明存档程序在创建存档副本时是否使用锁定的缓冲区。如果指定 lock，存档程序将在 sam-arcopy(1M) 操作期间在内存中的存档缓冲区上设置文件锁定。这可以避免由于为每一个 I/O 请求锁定和取消锁定缓冲区而造成的开销，从而减少占用系统 CPU 的时间。 仅在配有大量内存的大型系统上，才有必要指定 lock 变量。如果内存不足，则可能会导致内存用尽的情况。 只有已为需要存档的文件启用直接 I/O 时， lock 变量才有效。默认情况下，不会指定 lock 参数，文件系统会在所有直接 I/O 缓冲区上设置锁定（也包括用于存档的缓冲区）。有关启用直接 I/O 的详细信息，请参阅 setfa(1) 手册页、 sam_setfa(3) 库例程手册页，或 mount_samfs(1M) 手册页中介绍的 -O forcedirectio 选项。

例如，您可以按以下方式在 archiver.cmd 文件的指令行中指定该指令：

```
bufsize=od 7 lock
```

您可以使用 -bufsize 和 -lock 存档组副本参数，以存档组为单位设置缓冲区的大小和锁定。有关的详细信息，请参阅第 84 页的 “存档组副本参数”。

drives 指令：控制用于存档的驱动器数

默认情况下，存档程序使用自动化库中的所有驱动器进行存档。要限制存档程序使用的自动化库中的驱动器数量，请使用 drives 指令。

此指令的格式如下：

```
drives=auto_lib  count
```

表 4-5　drives 指令的变量

变量	含义
auto_lib	mcf 文件中定义的自动化库的系列集名。
count	用于存档活动的驱动器数量。

另请参阅第 85 页的“指定存档请求的驱动器数：-drivemax、 -drivemin 和 -drives”，其中介绍了 -drivemax、 -drivemin 和 -drives 存档组副本参数。

examine 指令：控制存档扫描

新文件以及已更改的文件都是备选的存档文件。存档程序会通过以下某一方法来查找此类文件：

- 连续存档。当进行连续存档时，存档程序对文件系统进行处理，检测其中刚刚发生更改的文件。
- 基于扫描的存档。而对于基于扫描的存档，存档程序则会定期扫描文件系统，寻找需要存档的文件。

examine 指令控制存档程序是执行连续存档，还是执行基于扫描的存档，具体如下：

```
examine=method
```

可以为 *method* 指定的关键字如表 4-6 所示。

表 4-6　examine 指令 *method* 变量的值

method 值	含义
noscan	指定连续存档。在执行初次扫描后，仅当目录内容发生更改以及需要进行存档时，才会对其进行扫描。但不对目录和索引节点信息进行扫描。这种存档方法的性能要好于基于扫描的存档，尤其是当文件系统中文件的数量大于 1,000,000 时。默认设置。
scan	指定基于扫描的存档。第一次的文件系统扫描是执行目录扫描。接下来对索引节点进行扫描。
scandirs	指定仅对目录执行基于扫描的存档。指定后，如果存档程序查找到具有 no_archive 属性的目录，将不对其进行扫描。这种目录中含有未更改的文件，跳过这种目录这可以大大缩短存档扫描时间。
scaninodes	指定仅对索引节点执行基于扫描的存档。

interval 指令：指定存档时间间隔

存档程序定期检查所有已安装的 Sun StorEdge SAM-FS 文件系统的状态。检查时间由存档时间间隔控制。*存档时间间隔* 是指在每一个文件系统上执行扫描操作的时间间隔。要更改时间间隔，请使用 interval 指令。

注意 – 只有在 archiver.cmd 文件中也指定了 examine=scan 指令，interval 指令才会有效。

此指令的格式如下：

```
interval=time
```

其中的 *time*，用于指定对文件系统执行扫描操作的时间间隔，单位为秒。默认情况下， *time* 以秒计。默认设置为 interval=600，即 10 分钟。也可以将时间单位指定为分钟、小时等等。有关指定时间单位的信息，请参阅第 63 页的表 4-2 “archiver.cmd 文件指令单位”。

如果存档程序接收到 samu(1M) 实用程序的 :arrun 命令，将会立即扫描所有文件系统。如果 archiver.cmd 文件中还指定了 examine=scan 指令，那么会在运行 :arrun 或 :arscan 后进行扫描。

如果为文件系统设置了 hwm_archive 安装选项，则可以自动缩短存档时间间隔。此安装选项指定存档程序在文件系统填满且超过空间占用上限时即开始进行扫描。high=*percent* 安装选项用于设置文件系统空间占用的上限。

有关指定存档时间间隔的详细信息，请参阅 archiver.cmd(4) 手册页。有关设置安装选项的详细信息，请参阅 mount_samfs(1M) 手册页。

logfile 指令：指定存档程序日志文件

存档程序可以生成一个日志文件，其中包含每一个存档、重新存档或自动取消存档的文件的有关信息。日志文件连续地记录存档操作。要指定日志文件，请使用 logfile 指令。此指令的格式如下：

```
logfile=pathname
```

其中的 *pathname* 用于指定日志文件的绝对路径和名称。默认情况下，系统不会生成此文件。

示例。假定您希望通过将前一天的日志文件复制到另一位置，以实现存档程序日志文件的每日备份。如果您确保在存档程序日志文件关闭时执行复制，则可以达到此目的。换而言之，在系统打开存档程序日志文件以向其中写入信息时，您不能执行复制操作。

▼ 备份存档程序日志文件

您需要执行以下步骤：

1. **使用 `mv`(1) 命令在 UFS 中移动存档程序日志文件。**

   这可给予 `sam-arfind`(1M) 或 `sam-arcopy`(1M) 一定的操作时间，以完成向存档程序日志文件写入信息。

2. **使用 `mv`(1) 命令将前一天的存档程序日志文件移至 Sun StorEdge SAM-FS 文件系统。**

   此外，您还可为单个文件系统设置 `logfile` 指令。

`notify` 指令：重命名事件通知脚本

`notify` 指令用于将存档程序的事件通知脚本文件的名称设置为 *filename*。此指令的格式如下：

```
notify=filename
```

其中的 *filename*，用于指定包含存档程序事件通知脚本的文件名称，或指定该文件的完整路径。

默认文件名如下所示：

```
/etc/opt/SUNWsamfs/scripts/archiver.sh
```

存档程序执行此脚本，来根据特定站点的要求处理各种事件。此脚本通过第一个变量的关键字进行调用。其关键字包括：`emerg`、`alert`、`crit`、`err`、`warning`、`notice`、 `info` 和 `debug`。

其它变量在默认脚本中加以说明。有关详细信息，请参阅 `archiver.sh`(1M) 手册页。

ovflmin 指令：控制卷溢出功能

卷溢出 是指允许在多个卷上存档文件的过程。在 archiver.cmd 文件中使用 ovflmin 指令即可启用卷溢出功能。当文件的大小超过 ovflmin 指令的 *minimum_file_size* 值时，存档程序会将文件的另一部分写入至另一个同类型的可用卷中 （如有必要）。写入至每一个卷的文件部分称为*片段*。

注意 – 使用卷溢出功能之前，请务必充分理解卷溢出的含义。只有在全面检验卷溢出对您的站点所产生的影响后，才可以使用卷溢出功能。文件存档在多个卷上会严重地加大故障恢复操作和回收操作的难度。

存档程序通过 ovflmin 指令来控制卷溢出功能。ovflmin 指令用于指定允许文件溢出卷的最小大小。默认情况下，存档程序会禁用卷溢出功能。

此指令的格式如下：

```
ovflmin = media  minimum_file_size
```

表 4-7 ovflmin 指令的变量

变量	含义
media	介质类型。有关有效介质类型的列表，请参阅 mcf(4) 手册页。
minimum_file_size	指定文件溢出时的最小大小。

示例 1。假设许多文件存档在 mo 介质卡盒中，并且产生严重的碎片 （例如 25%）。由于这些文件不能占用卷的整个空间，因而导致每个卷中存有大量的未用空间。为了更好地利用卷内空间，您需要将 mo 介质的 ovflmin 设置成略小于最小文件大小的值。下面的指令将其设置为 150 MB：

```
ovflmin=mo 150m
```

请注意，在本示例中启用卷溢出功能会造成在存档和登台文件时需要载入两个卷。

此外，您还可为单个存档组设置 ovflmin 指令。

示例 2。 sls(1) 命令可列出存档副本，并显示每个 VSN 上文件的每个片断。代码示例 4-4 显示了一个文件的存档程序日志文件以及 sls -D 命令的输出，这个文件名为 file50，它是一个存档在多个卷上的大型文件。

代码示例 4-4 存档程序日志文件示例

```
A  97/01/13  16:03:29  lt  DLT000  big.1  7eed4.1  samfs1  13.7
477609472  00 big/file50  0  0

A  97/01/13  16:03:29  lt  DLT001  big.1  7fb80.0  samfs1  13.7
516407296  01 big/file50  0  1

A  97/01/13  16:03:29  lt  DLT005  big.1  7eb05.0  samfs1  13.7
505983404  02 big/file50  0  2
```

代码示例 4-4 显示了 file50 存档在三个卷上，其 VSN 分别为 DLT000、DLT001 和 DLT005。每个片段在卷上的位置和大小分别显示在第七个和第十个字段中，它们的值与另外显示的 sls -D 输出中的值相匹配。有关存档日志条目的详细说明，请参阅 archiver(1M) 手册页。

代码示例 4-5 显示了 sls -D 命令及其输出。

代码示例 4-5 sls(1M) -D 命令及其输出

```
# sls -D file50
file50:
  mode: -rw-rw----  links:   1  owner: gmm    group: sam
  length: 1500000172  admin id:      7  inode: 1407.5
  offline;   archdone;  stage -n
  copy1: ---- Jan 13 15:55            lt
    section 0:  477609472     7eed4.1    DLT000
    section 1:  516407296     7fb80.0    DLT001
    section 2:  505983404     7eb05.0    DLT005
  access:      Jan 13 17:08  modification: Jan 10 18:03
  changed:     Jan 10 18:12  attributes:   Jan 13 16:34
  creation:    Jan 10 18:03  residence:    Jan 13 17:08
```

卷溢出文件不能生成校验和。有关使用校验和的详细信息，请参阅 ssum(1) 手册页。

注意 – 请记住，当使用卷溢出功能时，一旦出现故障，恢复卷溢出数据是非常困难的。有关如何恢复这类文件的信息，请参阅 *《Sun QFS、 Sun SAM-FS 和 Sun SAM-QFS 故障恢复指南》*中的示例。有关详细信息，请参阅 request(1) 手册页。

---

wait 指令：推迟存档程序的启动

wait 指令命令存档程序等待来自 samu(1M) 或 SAM-QFS Manager 的启动信号。当收到此信号后，存档程序才开始典型的存档操作。默认情况下，存档程序在由 sam-fsd(1M) 启动后即开始存档操作。要推迟存档操作，请使用 wait 指令。此指令的格式如下：

```
wait
```

此外，您还可为单个文件系统设置 wait 指令。

文件系统指令

可在 archiver.cmd 文件的全局指令后面添加专用于特定文件系统的 fs= 指令。遇到 fs= 指令后，存档程序假定后面的所有指令均仅适用于特定的文件系统。

fs 指令：指定文件系统

默认情况下，存档控制指令适用于所有文件系统。不过，您可以将某些控制指令的适用范围限定在特定的文件系统上。要指定特定的文件系统，可使用 fs 指令。此指令的格式如下：

```
fs=fsname
```

其中的 *fsname*，用于指定在 mcf 文件中定义的文件系统名。

这些指令后面出现的全局指令和存档组相关指令仅适用于指定的文件系统，直到出现下一个 fs= 指令。例如，您可以使用此指令为每一个文件系统指定不同的日志文件。

其它文件系统指令

有几个指令既可以指定为适用于所有文件系统的全局指令，也可以指定为专用于某一文件系统的指令。无论在何处指定，它们的效用都是相同的。这几个指令如下：

- interval 指令。有关此指令的详细信息，请参阅第 69 页的 “interval 指令：指定存档时间间隔”。
- logfile 指令。有关此指令的详细信息，请参阅第 69 页的 “logfile 指令：指定存档程序日志文件”。

- `wait` 指令。有关此指令的详细信息，请参阅第 73 页的“wait 指令：推迟存档程序的启动”。

存档组分配指令

默认情况下，文件将作为根据文件系统命名的存档组的一部分进行存档。在 SAM-QFS Manager 中，*存档策略* 用于定义存档组。不过，您可以指定其它存档组以包含具有类似特征的文件。如果某个文件不属于您指定的任何存档组，则它将作为根据文件系统命名的默认存档组的一部分进行存档。

分配存档组

存档组成员指令可以将具有类似特征的文件分配至同一个存档组。这些指令的语句位于 `find`(1) 命令后面。每一个存档组分配指令均采用以下格式：

```
archive_set_name  path  [search_criteria1  search_criteria2  ...  ]  [file_attributes]
```

表 4-8　存档组分配指令的变量

变量	含义
archive_set_name	您的站点为存档组定义的名称。它必须是存档组分配指令中的第一个字段。存档组名通常暗示属于该存档组的文件的特征。指定存档组名时，只能使用 26 个英文字母、数字 (0-9) 和下划线字符 (_)，而不得使用其它特殊字符或空格。存档组名的第一个字符必须是字母。要阻止对某些文件进行存档，请将 *archive_set_name* 指定为 `no_archive`。
path	相对于文件系统安装点的路径。这使得存档组成员指令可适用于多个 Sun StorEdge SAM-FS 文件系统。如果要使该路径包括文件系统中的所有文件，请在路径字段中输入句点 (.)。不允许在路径的开头使用斜杠 (/)。*path* 所指定目录中的文件及其子目录均视为属于该存档组。
search_criteria1 *search_criteria2*	可以指定零个、一个或多个 *search_criteria* 变量。指定搜索标准的目的是根据文件大小、文件所有权以及其它要素限制存档组的范围。有关可用的 *search_criteria* 变量的信息，请参阅以下几节。
file_attributes	可以指定零个、一个或多个 *file_attributes* 参数。当 `sam-arfind` 进程在存档期间扫描文件系统时，将为文件设置这些文件属性。

示例 1。代码示例 4-6 显示了典型的存档组成员指令。

代码示例 4-6　存档组成员指令

```
hmk_files     net/home/hmk      -user hmk
datafiles     xray_group/data   -size 1M
system        .
```

示例 2。将文件归入名为 no_archive 存档组中，可以阻止存档程序对这些文件进行存档。代码示例 4-7 显示了阻止对 tmp 目录的各级子目录下的文件进行存档的指令行，而且不管 tmp 目录在文件系统中的位置如何，都是如此。

代码示例 4-7　阻止存档的存档指令

```
fs = samfs1
no_archive tmp
no_archive . –name .*/tmp/
```

以下几节介绍了可以指定的 *search_criteria* 参数。

文件大小 *search_criteria*：-access

可使用 -access *age* 特征来指定用于确定存档组成员的文件时限。如果使用这个 *search_criteria*，访问时间早于 *age* 的文件将被重新存档至其它介质中。其中的 *age*，用于指定一个整数，这个整数的后缀是表 4-9 中所示的某个字母。

表 4-9　-access *age* 的后缀

字母	含义
s	秒
m	分钟
h	小时
d	天
w	周
y	年

例如，可使用这个指令来指定将长时间未使用的文件重新存档至一个较为便宜的介质中。

文件大小 *search_criteria*：-minsize 和 -maxsize

可使用 -minsize *size* 和 -maxsize *size* 特征，通过文件的大小来确定存档组成员。其中的 *size*，用于指定一个整数，这个整数的后缀是表 4-10 中所示的某一字母。

表 4-10　-minsize 和 -maxsize *size* 的后缀

字母	含义
b	字节
k	千字节 (KB)
M	兆字节 (MB)
G	千兆字节 (GB)
T	兆兆字节 (TB)
P	千兆兆字节 (PT)
E	兆兆兆字节 (ET)

示例。本示例中的命令行指定所有介于 500 KB 和 100 MB 之间的文件均属于存档组 big_files。超过 100 MB 的文件属于存档组 huge_files。代码示例 4-8 显示了这些命令行。

代码示例 4-8　-minsize 和 -maxsize 指令的使用示例

```
big_files  .  -minsize 500k  -maxsize 100M
huge_files .  -minsize 100M
```

所有者和组 *search_criteria*：-user 和 -group

可使用 -user *name* 和 -group *name* 特征，通过所有权和组关系来确定存档组成员。代码示例 4-9 显示了这些指令的示例。

代码示例 4-9　使用 -user 和 -group 指令的示例

```
adm_set    .   -user sysadmin
mktng_set  .   -group marketing
```

所有属于用户 sysadmin 的文件均属于存档组 adm_set，所有组名称为 marketing 的文件均属于存档组 mktng_set。

使用样式匹配的文件名 *search_criteria*：-name *regex*

可以使用正则表达式来指定某个存档组中文件的名称。作为一项 *search_criteria*，-name *regex* 参数指定任何与正则表达式 *regex* 相匹配的完整路径均为存档组的成员。

regex 变量遵守 regexp(5) 手册页中列出的约定。请注意，正则表达式不遵守与 UNIX 通配符相同的约定。

在系统内部，所有位于选定目录之下的文件 （及其相对于文件系统安装点的指定路径）均会被列出并一起传递，以进行样式匹配。这使得您可以在 -name *regex* 字段中创建样式，以匹配文件名和路径名。

示例

1. 下面的指令将存档组 images 中的文件限定为那些以 .gif 结尾的文件：

```
images  . -name \.gif$
```

2. 下面的指令选择以字符 GEO 开头的文件：

```
satellite  . -name /GEO
```

3. 您可以将正则表达式与 no_archive 存档组结合使用。下面的指令可阻止对任何以 .o 结尾的文件进行存档：

```
no_archive  . -name \.o$
```

4. 假定您的 archiver.cmd 文件中包含代码示例 4-10 中所示的指令行：

代码示例 4-10　正则表达式示例

```
# File selections.
fs = samfs1
     1 1s
     2 1s
no_archive share/marketing -name fred\.
```

对于此 archiver.cmd 文件，存档程序不会对用户目录或其子目录下的 fred.* 文件进行存档。文件的存档情况如下所述：

■ 如果指定了如代码示例 4-10 中所示的指令，那么将不对代码示例 4-11 中所示的文件进行存档。

代码示例 4-11 不进行存档的文件 （假定所指定的指令如代码示例 4-10 所示）

```
/sam1/share/marketing/fred.anything
/sam1/share/marketing/first_user/fred.anything
/sam1/share/marketing/first_user/first_user_sub/fred.anything
```

■ 如果指定了如代码示例 4-10 中所示的指令，那么将对代码示例 4-12 中所示的文件进行存档。

代码示例 4-12 进行存档的文件 （假定所指定的指令如代码示例 4-10 所示）

```
/sam1/fred.anything
/sam1/share/fred.anything
/sam1/testdir/fred.anything
/sam1/testdir/share/fred.anything
/sam1/testdir/share/marketing/fred.anything
/sam1/testdir/share/marketing/second_user/fred.anything
```

5. 假定您的 `archiver.cmd` 文件中包含代码示例 4-13 中所示的指令行。

代码示例 4-13 `archiver.cmd` 文件示例

```
# File selections.
fs = samfs1
     1 1s
     2 1s
no_archive share/marketing -name ^share/marketing/[^/]*/fred\.
```

代码示例 4-13 中的 `archiver.cmd` 文件不会对用户根目录下的 `fred.*` 文件进行存档。但会对用户子目录和 `share/marketing` 目录下的 `fred.*` 文件进行存档。在此情况下，用户根目录刚好是 `first_user`。本示例中，从 `share/marketing/` 到下一个斜杠字符 (/) (/) 之间的任何目录均被视为用户的根目录。文件的存档情况如下所述：

■ 如下所示的文件不会被存档：

```
/sam1/share/marketing/first_user/fred.anything
```

■ 如果指定了如代码示例 4-13 中所示的指令，那么将对代码示例 4-14 中所示的文件进行存档。

代码示例 4-14 进行存档的文件 （假定所指定的指令如代码示例 4-13 所示）

```
/sam1/share/fred.anything
/sam1/share/marketing/fred.anything
/sam1/share/marketing/first_user/first_user_sub/fred.anything
/sam1/fred.anything
/sam1/testdir/fred.anything
/sam1/testdir/share/fred.anything
/sam1/testdir/share/marketing/fred.anything
/sam1/testdir/share/marketing/second_user/fred.anything
/sam1/testdir/share/marketing/second_user/sec_user_sub/fred.any
```

释放和登台 *file_attributes*：-release 和 -stage

您可以通过分别使用 -release 和 -stage 选项，设置与存档组内文件相关联的释放和登台属性。这两种设置都会取代用户以前可能设置的释放或登台属性。

-release 选项的格式如下所示：

```
-release attributes
```

-release 指令的 *attributes* 所遵守的约定与 release(1) 命令相同，具体如表 4-11 所示。

表 4-11 -release 选项

属性	含义
a	完成第一个存档副本后，释放文件占用的磁盘空间。
d	复位为默认设置。
n	永不释放文件占用的磁盘空间。
p	释放文件占用的部分磁盘空间。

-stage 选项的格式如下所示：

```
-stage attributes
```

-stage 指令的 *attributes* 所遵守的约定与 stage(1) 命令相同，具体如表 4-12 所示。

表 4-12　-stage 指令的 *attributes*

属性	含义
a	联合登台本存档组中的文件。
d	复位为默认设置。
n	永远不登台本存档组中的文件。

下面的示例显示了如何使用文件名参数和文件属性，来部分释放 Macintosh 资源目录：

```
MACS   . -name .*/\.rscs/   -release p
```

存档组成员冲突

有时，在选择路径和其它文件特征以将文件归入存档组时，可能会造成存档组成员混乱。系统采用下列方式来解决此类情况：

1. 选择存档组中最初的成员定义。

2. 首先选择文件系统本地的成员定义，然后选择全局性的成员定义。

3. 如果某个成员定义与先前的成员定义完全相同，则系统会发出错误通知。

因此，用户应在指令文件的前部设置更加严格的成员定义。

如果对特定文件系统的存档操作进行了控制 （使用 fs=*fsname* 指令），那么存档程序在检验全局指令之前，会先检验文件系统的特定指令。这样，文件可以分配至本地存档组 （包括 no_archive 存档组），而不是分配至全局存档组。在设置 no_archive 等全局存档组分配时，就默认采用此规则。

代码示例 4-15 显示了一个 archiver.cmd 文件。

代码示例 4-15　可能存在成员冲突的 archiver.cmd 文件

```
no_archive    . -name .*\.o$
fs = samfs1
     allfiles   .
fs = samfs2
     allfiles   .
```

从代码示例 4-15 可以看出，管理员本不希望对以上两个文件系统中的 .o 文件进行存档。但是，由于对本地存档组分配 allfiles 的检验先于全局存档组分配 no_archive，因此仍会对 samfs1 和 samfs2 文件系统中的 .o 文件进行存档。

代码示例 4-16 显示了确保能够不对以上两个文件系统中的 .o 文件进行存档的指令。

代码示例 4-16 更正后的 archiver.cmd 文件

```
fs = samfs1
     no_archive    . -name .*\.o$
     allfiles   .
fs = samfs2
     no_archive    . -name .*\.o$
     allfiles   .
```

存档副本指令

如果不指定存档副本份数，则存档程序将为存档组中的文件创建一份存档副本。默认情况下，创建文件副本的存档时限为四分钟。如果您需要多份副本，则必须使用存档副本指令指定所有副本（包括第一个副本）。

存档副本指令以一个整数 *copy_number* 开头。这个数字（1、2、3 或 4）是副本份数。数字后面是一个或多个用于指定该副本存档特征的变量。

存档副本指令必须紧跟在与它们相关的存档组分配指令的后面。每一个存档副本指令均采用以下格式：

```
copy_number [ -release | -norelease ] [archive_age] [unarchive_age]
```

以下几节介绍这些存档副本指令变量。

存档之后释放磁盘空间：-release

可以在副本份数后面使用 -release 指令，来指定在创建存档副本之后自动释放的文件所占用的磁盘空间。此选项的格式如下：

```
-release
```

在代码示例 4-17 中，属于组 images 的文件在其存档时限达到 10 分钟时进行存档。在创建第一个存档副本后，系统会释放文件占用的磁盘高速缓存空间。

代码示例 4-17 使用了 -release 指令的 archiver.cmd 文件

```
ex_set . -group images
    1 -release 10m
```

推迟释放磁盘空间：-norelease

您可能希望在创建多份存档副本之后再释放文件占用的磁盘空间。那么可以使用 -norelease 选项，它将使得系统在所有标记了 -norelease 的副本创建完毕之后，才自动释放磁盘高速缓存。此选项的格式如下：

```
-norelease
```

代码示例 4-18 指定了一个名为 vault_tapes 的存档组。系统将为其创建两份副本，且只有在这两份副本创建完毕之后，才释放与该存档组相关的磁盘高速缓存。如果站点在创建离站存储的卷之前需要对文件进行在线访问，那么可以采用此方案。

代码示例 4-18 使用了 -norelease 指令的 archiver.cmd 文件

```
vault_tapes
    1 -norelease 10m
    2 -norelease 30d
```

请注意，仅需创建单个存档副本时 -norelease 参数对自动释放磁盘空间无效，因为至少在创建一个副本后才会释放磁盘空间。另外， -norelease 和 -release 这两个参数相互排斥，不能同时使用。

设置存档时限

可以通过在该指令的下一个字段中指定存档时限来设置文件的存档时限。存档时限的后缀字符可以指定为 h （小时）或 m （分钟）。第 63 页的表 4-2 “archiver.cmd 文件指令单位”详细列出了这些后缀字符及其含义。

在代码示例 4-19 中，目录 data 中的文件将在其存档时限到达 1 个小时后进行存档。

代码示例 4-19 指定了存档时限的 archiver.cmd 文件

```
ex_set data
    1 1h
```

自动取消存档

如果您为文件指定多个存档副本，则可能需要只保留一个存档副本，而自动取消对其它所有副本进行存档。当使用不同存档时限将文件存档至不同介质时，可能发生此类情况。

代码示例 4-20 显示了指定取消存档时限的指令。

代码示例 4-20 指定了取消存档时限的 `archiver.cmd` 文件

```
ex_set home/users
     1 6m 10w
     2 10w
     3 10w
```

路径 `home/users` 中文件的第一个副本将在文件修改后六分钟进行存档。当文件的存档时间达到 10 周时，系统会创建第二个和第三个存档副本，并且会取消对第一个副本进行存档。

有关控制取消存档的其它方法，请参阅第 90 页的 “控制取消存档”。

为元数据指定多份副本

如果需要多份元数据副本，那么可以在指令文件中放置副本份数定义，其位置是紧跟在 `fs=` 指令后。

代码示例 4-21 显示了一个指定了多份元数据副本的 `archiver.cmd` 文件。

代码示例 4-21 指定了多份元数据副本的 `archiver.cmd` 文件

```
fs = samfs7
     1 4h
     2 12h
```

在本示例中，系统将在四小时后创建 `samfs7` 文件系统的元数据的第一份副本，然后在 12 小时后创建第二份副本。

文件系统元数据包括对文件系统中路径名的更改。因此，如果您经常更改目录，则系统会创建新的存档副本。这会造成系统频繁地载入您为元数据指定的卷。

存档组副本参数

archiver.cmd 文件中，存档组参数部分以 params 指令开头，以 endparams 指令结尾。代码示例 4-22 给出了存档组指令的格式。

代码示例 4-22 存档组副本参数的格式

```
params
archive_set_name.copy_number[R] [ -param1 -param2 ...]
.
.
.
endparams
```

表 4-13 存档组副本参数的变量

变量	含义
archive_set_name	您的站点为存档组定义的名称。通常可暗示属于该存档组的文件的特征。可以是 allsets。指定存档组名时，只能使用 26 个英文字母、数字 (0-9) 和下划线字符 (_)，而不得使用其它特殊字符或空格。存档组名的第一个字符必须是字母。
.	句点 (.) 字符。用于分隔 *archive_set_name* 和 *copy_number*。
copy_number	用于定义存档副本份数的整数。可以是 1、 2、 3 或 4。
R	指定所定义的参数是用于定义此存档组要重新存档的副本数。例如，可以在 *-param1* 变量中，通过使用 R 和指定 VSN 将重新存档的副本定向到特定的卷。
-param1 *-param2*	一个或多个参数。以下几小节介绍 params 和 endparams 指令之间可以指定的参数。

伪存档组 allsets 提供了一种为所有存档组设置默认存档组指令的方法。所有 allsets 指令必须位于实际存档组副本指令的前面，其原因是为单个存档组副本设置的参数可以取代由 allsets 指令设置的参数。有关 allsets 存档组的详细信息，请参阅 archiver.cmd(4) 手册页。

本节介绍除 -disk_archive 参数之外的其它所有存档组处理参数。有关 -disk_archive 参数的详细信息，请参阅第 101 页的 “磁盘存档”。

控制存档文件的大小：-archmax

-archmax 指令用于指定存档组文件的最大大小。格式如下：

```
-archmax target_size
```

这条指令与 archmax 全局指令非常相似。有关这条指令以及 *target_size* 的输入值的信息，请参阅第 66 页的 “archmax 指令：控制存档文件的大小”。

设置存档程序缓冲区大小：-bufsize

默认情况下，需要存档的文件先存储在缓冲区的内存中，然后再写入存档介质。可以使用 -bufsize 参数指定非默认的缓冲区大小。这些操作可以改善系统的性能。您可以尝试不同的 *buffer_size* 值以确定最适合的缓冲区大小。

该参数的格式如下：

```
-bufsize=buffer_size
```

其中的 *buffer_size*，可用于指定一个 2 至 32 的值。默认值为 4。这个值乘以该介质类型的 *dev*_blksize 值，计算结果即为所使用的缓冲区大小。*dev*_blksize 的值可以在 defaults.conf 文件中指定。有关此文件的详细信息，请参阅 defaults.conf(4) 手册页。

例如，可以按以下方式在 archiver.cmd 文件的命令行中指定该参数：

```
myset.1 -bufsize=6
```

此外，通过指定全局 bufsize=*media buffer_size* 指令，也可以达到与该指令同样的效果。有关此主题的详细信息，请参阅第 66 页的 “bufsize 指令：设置存档程序缓冲区大小”。

指定存档请求的驱动器数：-drivemax、 -drivemin 和 -drives

默认情况下，存档程序只使用一个介质驱动器对存档组中的文件进行存档。如果存档组中的文件数量众多或比较大，则使用多个驱动器可以帮助您更有效地存档。另外，如果自动化库中驱动器的运行速度不同，使用这些指令可以使存档更为高效。

代码示例 4-23 显示了可用于将存档请求拆分至多个驱动器，以及平衡磁带驱动器传输速度差异的参数。

代码示例 4-23 `-drivemax`、`-drivemin` 和 `-drives` 指令的格式

```
-drivemax max_size
-drivemin min_size
-drives number
```

表 4-14 `-drivemax`、`-drivemin` 和 `-drives` 参数的变量

变量	含义
maxsize	一个驱动器中可存档的最大数据量。
minsize	一个驱动器中可存档的最小数据量。其默认值是 `-archmax` *target_size* 的值 （如果已指定）或是该介质类型的默认值。 如果指定了 `-drivemin` *minsize* 参数，则 Sun StorEdge SAM-FS 只在数据量足够大时才使用多个驱动器。作为一个指导原则，可以将 *minsize* 设置为足够大，以使传输时间远远大于卡盒更改时间（载入、定位、卸载）。
number	用于对此存档组进行存档的驱动器数量。默认设置为 1。

系统会根据所指定的参数来检验存档请求，具体如下：

- 如果存档请求小于 *min_size*，则只使用一个驱动器来写入存档请求。
- 如果存档请求大于 *min_size*，则根据 *min_size* 检验存档请求，并安排适当数量的驱动器，但最多不超过指定的最大驱动器数。
- 如果 *min_size* 为零，则尝试将存档请求中的文件存档在所指定最大数量的驱动器上。

当使用 `-drives` 参数时，仅在一次存档的数据量大于 *min_size* 时，才使用多个驱动器。实际并行使用的驱动器数是 *arch_req_total_size*/*min_size* 和 `-drives` 参数指定的驱动器数这两者当中的较小者。

要设置这些参数，用户需要考虑文件的创建速度、载入和卸载驱动器所需的时间以及驱动器的传输速率。

示例 1。如果希望将存档请求分割至各个驱动器，但是您又不希望将较小的存档请求也分割在全部驱动器上，则可以使用 `-drivemin` 和 `-drives` 参数。这适用于大型文件的操作。

示例 2。假设您在五个驱动器上分割一个名为 `big_files` 的存档组。这时，此存档组可如表 4-15 中所示进行分割，但具体分割方法取决于存档组的大小。

表 4-15 存档组分割示例

存档组大小	驱动器数
< 20 GB	1
≥ 20 GB 至 < 30 GB	2
≥ 30 GB 至 < 40 GB	3
≥ 40 GB 至 < 50 GB	4
≥ 50 GB	5

代码示例 4-24 显示了 `archiver.cmd` 文件中用于将存档请求分割至多个驱动器的指令行。

代码示例 4-24 用于将存档请求分割至多个驱动器的指令

```
params
bigfiles.1 -drives 5 -drivemin 10G
endparams
```

示例 3。下行是在 `archiver.cmd` 文件中指定的：

```
huge_files.2 -drives 2
```

当存档组 `huge_files.2` 中的文件总大小等于或大于介质的 `drivemin` 的两倍时，使用两个驱动器来存档文件。

示例 4。存档请求的大小为 300 GB。下面是 `archiver.cmd` 文件中的指令行，它指定每次在这五个驱动器中的一个驱动器上存档 10 GB 的文件：

```
-drives 5 -drivemax 10G
```

最大化卷上的空间：`-fillvsns`

默认情况下，存档程序在写入存档副本时，将使用分配给存档组的所有卷。而且存档程序在写入存档副本时，会选择一个空间足以容纳所有文件的卷。这将导致卷不会被完全占用。如果指定了 `-fillvsns`，存档程序则将存档请求拆分为一个较小的组。

设置存档缓冲区锁定：-lock

默认情况下，需要存档的文件先存储在缓冲区的内存中，然后再写入存档介质。如果已启用直接 I/O，则可以使用 -lock 参数锁定此缓冲区。此操作可以改善系统的性能，并且可以尝试不同的参数值，以达到最佳性能。

此参数的格式如下：

```
-lock
```

-lock 参数指明存档程序在创建存档副本时是否使用锁定的缓冲区。如果指定 -lock，存档程序将在 sam-arcopy(1M) 操作期间，在内存中的存档缓冲区上设置文件锁定。这样可避免对缓冲区进行分页，因而提高了系统性能。

仅在配有大量内存的大型系统上，才有必要指定 -lock 参数。如果内存不足，则可能会导致内存用尽。

只有已为需要存档的文件启用直接 I/O 时，-lock 参数才有效。默认情况下，不会指定 -lock 参数，并且文件系统会在所有直接 I/O 缓冲区上设置锁定（包括用于存档的缓冲区）。有关启用直接 I/O 的详细信息，请参阅 setfa(1) 手册页、 sam_setfa(3) 库例程手册页，或 mount_samfs(1M) 手册页中介绍的 -O forcedirectio 选项。

例如，可以按以下方式在 archiver.cmd 文件的命令行中指定该参数：

```
yourset.3 -lock
```

另外，还可通过指定 bufsize=*media buffer_size* [lock] 指令的 lock 变量，来指定全局性的、与该参数等效的参数。有关此主题的详细信息，请参阅第 66 页的“bufsize 指令：设置存档程序缓冲区大小”。

创建离线文件的存档副本：-offline_copy

为文件创建了一份存档副本后，此文件即成为可释放的备选文件。如果在创建所有存档副本前，释放文件并使其处于离线状态，那么存档程序会使用此参数来确定创建其他存档副本时所应使用的方法。在选择所使用的 *method* 时，要考虑 Sun SAM-FS 系统的可用驱动器数以及可用的磁盘高速缓存空间。此参数的格式如下：

```
-offline_copy method
```

为 *method* 指定如下表所示的某个关键字。

表 4-16 `-offline_copy` 指令中 *method* 变量的值

方法	含义
`none`	在将文件复制到存档卷中之前，根据需要登台每个文件。默认设置。
`direct`	不使用高速缓存，而是将文件从离线卷直接复制到存档卷。此方法假定来源卷和目标卷是不同的卷，而且有两个可用驱动器。如果指定了此方法，则需将 `stage_n_window` 安装选项的值增大，使其大于 256 KB 的默认值。有关安装选项的详细信息，请参阅 `mount_samfs`(1M) 手册页。
`stageahead`	在对一个文件进行存档的同时，登台另一个文件。如果指定了这种方法，系统会在将某文件写入目标位置的同时，登台下一个存档文件。
`stageall`	在存档前，将所有文件登台到磁盘高速缓存。这种方法仅使用一个驱动器，并假定磁盘高速缓存的空间足以容纳所有文件。

指定回收

回收进程可用于收回由过期存档映像所占用的存档卷中的空间。默认情况下，不进行回收。

要进行回收，可在 `archiver.cmd` 文件和 `recycler.cmd` 文件中设置指令。有关 `archiver.cmd` 文件中支持的回收指令的详细信息，请参阅第 159 页的“回收”。

联合存档：`-join`

如果指定了 `-join path` 参数，存档程序将使用联合存档。如果希望将整个目录存档至一个卷，并且知道存档文件实际只占用一个卷，则使用联合存档可以满足您的要求。否则，如果希望将目录保存在一起，请使用 `-sort path` 或 `-rsort path` 参数将文件连续地保存在一起。`-rsort` 可执行逆向排序。

当存档程序将存档文件写入卷时，它会使用用户文件有效地压缩卷。将来，当从同一目录中访问文件时，会出现一定的延迟，因为登台进程需要对卷进行重新定位以读取下一个文件。为了缩短延迟，可以将来自同一目录路径的文件连续地存档至一个存档文件中。联合存档进程将取代空间效率运算法则，从而将来自同一目录的文件存档在一起。`-join path` 参数可以使这些文件在存档组副本中连续地存档在一起。

当无需更改文件内容但始终需要同时访问一组文件时，使用联合存档可以满足此类要求。例如在医院里，用户可能使用联合存档方法来访问医疗图片。与同一位患者关联的图片可以保存在同一个目录中，以便医生可以同时访问这些图片。如果根据静态图片的目录位置连续存档图片，则可以更为方便、快捷地访问它们。例如：

```
patient_images.1 -join path
```

注意 – `-join path` 参数可将来自相同目录的数据文件写入至同一个存档文件。如果目录很多，但其中的文件较小且数量不多，存档程序会创建很多较小的存档文件。由于数据文件对每个存档文件的 `tar(1)` 头文件来说相对较小，因此这些较小的分散文件会降低系统的写入性能。这会降低高速磁带驱动器的写入性能。

另外，由于 `-join path` 参数指定将来自同一目录中的所有文件都存档在单一卷中，因此有可能找不到合适的可用卷。在这种情况下，如果不为存档组分配更多的卷，这些文件将无法存档。另一种可能是一组要存档的文件太大，以至于永远不能存档在单个卷上。在这种情况下，这些文件永远无法存档。

对于大多数应用场合而言，如果不需要对 `-join path` 操作进行更为严格的限制，那么 `-sort path` 或 `-join path` 参数是最佳选择。

另外，也可以按文件存档时限、大小或路径对存档组副本中的文件进行排序。`age` 和 `size` 变量互相排斥，不可同时使用。代码示例 4-25 显示了如何使用 `-sort` 参数以及 `age` 或 `size` 变量，来对存档组进行排序。

代码示例 4-25 对存档组进行排序的指令

```
cardiac.1 -sort path
cardiac.2 -sort age
catscans.3 -sort size
```

第一行强制存档程序按路径名对存档请求进行排序。第二行强制存档程序根据文件的存档时限，按从旧到新的顺序，对名为 `cardiac.2` 的存档组副本进行排序。第三行强制存档程序根据文件的大小，按照从小到大的顺序，对名为 `catscans` 的存档组副本进行排序。如果希望逆向排序，可以用 `-rsort` 替代 `-sort`。

控制取消存档

取消存档 是指删除文件或目录的存档条目的过程。默认情况下，文件永远不会被取消存档。存档程序根据上一次访问文件的时间来确定是否取消存档。所有经常访问的数据都可以存储在磁盘等快速介质中，而其它所有不经常访问的旧数据可以存储在磁带中。

示例 1。代码示例 4-26 显示了一个 `archiver.cmd` 文件。

代码示例 4-26　控制取消存档的指令

```
arset1 dir1
   1 10m    60d
   2 10m
   3 10m
vsns
arset1.1    mo        OPT00[0-9]
arset1.2    lt        DLTA0[0-9]
arset1.3    lt        DLTB0[0-9]
```

如果代码示例 4-26 所示的 `archiver.cmd` 文件所控制的文件常被访问，它将始终保留在磁盘中，即使它的存档时间超过 60 天。只有在该文件未被访问的时间超过 60 天时才会删除副本 1 信息。

如果存档程序因文件未被访问的时间超过 60 天而删除其副本 1 信息，则当从副本 2 登台文件时，需要从磁带中读取文件。该文件恢复在线后，存档程序会在磁盘上为其创建新的副本 1，并且由此时开始计算为期 60 天的访问周期。如果该文件再次被访问，则 Sun StorEdge SAM-FS 存档程序会为其重新生成新的副本 1。

示例 2

假设某位患者在医院里进行为期四周的治疗。在此期间，该患者的所有文件位于快速介质中 （副本 1 = `mo`）。四周后，该患者出院。在该患者出院后的 60 天内，如果该患者的数据从未被访问过，则会取消存档索引节点中的副本 1 条目，而只保留副本 2 和副本 3 条目。此时，用户可以回收卷以便为新患者腾出空间，从而避免增加磁盘库。如果该患者六个月后到医院复查，则首先从磁带 （副本 2）访问数据。现在，存档程序会在磁盘中自动创建一个新的副本 1，以确保在复查期间 （可能持续数天或数周）可以从快速介质中访问数据。

控制存档文件的写入方式：`-tapenonstop`

默认情况下，存档程序会在存档文件之间写入一个磁带标记 （EOF 标签）和多个磁带标记。当启动下一个存档文件时，驱动程序会返回到第一个磁带标记后面的位置，因而会造成性能降低。`-tapenonstop` 参数可以指示存档程序只写入初始的磁带标记。这样，驱动程序只需返回到上一个磁带标记 （而不是第一个磁带标记）后面的位置，因而提高了性能。另外，如果指定 `-tapenonstop` 参数，存档程序将在复制操作的最后输入存档信息。

有关 `-tapenonstop` 参数的详细信息，请参阅 `archiver.cmd`(4) 手册页。

保留卷：-reserve

默认情况下，存档程序会将存档组副本写入由 archiver.cmd 文件中卷关联部分所述的正则表达式所指定的卷。但是，有时可能需要存档组卷只包含来自同一存档组的文件。保留卷进程可以满足这一数据存储要求。

注意 – -reserve 参数用于保留专供一个存档组使用的卷。站点使用保留卷时，可能导致频繁的卡盒载入和卸载操作。

-reserve 参数可为存档组保留卷。在设置 -reserve 参数，并将一个卷分配给某个存档组副本后，该卷的标识不会分配至其它任何存档组副本，即使某个正则表达式与之相匹配。

选择供某个存档组使用的卷后，系统会为该卷分配一个保留名称。此保留名称是联结存档组和卷的唯一标识。

-reserve 参数的格式如下所示：

```
-reserve keyword
```

此处指定的 *keyword* 取决于您所使用的格式。可用的格式包括存档组格式、所有者格式和文件系统格式，具体如下：

- 存档组格式。此格式使用 set *关键字* 的方式是：-reserve set
- 所有者格式。此格式使用以下某一*关键字*：dir、 user 或 group。代码示例 4-27 显示了这些指令的格式。

代码示例 4-27 -reserve 参数的所有者格式

```
-reserve dir
-reserve user
-reserve group
```

  代码示例 4-27 中所示的三种所有者格式相互排斥。也就是说，只能对存档组和副本使用这三种所有者格式的其中一种。

- 文件系统格式。此格式使用 fs *关键字* 的方式是：-reserve fs

在 archiver.cmd 文件中，您可为一种、两种或全部三种格式指定 -reserve 参数。在存档组参数定义中，这三种格式可以组合使用。

例如，代码示例 4-28 中显示了 archiver.cmd 文件的一个片断。以 arset.1 开头的行根据存档组、组和文件系统创建了一个保留名。

代码示例 4-28 指定了保留卷的 archiver.cmd 文件

```
params
arset.1 -reserve set -reserve group -reserve fs
endparams
```

保留卷的有关信息存储在库目录中。库目录中的行包括介质类型、 VSN、保留信息和保留日期及时间。保留信息包括存档组、路径名和文件系统三个部分，它们之间由双斜杠 (//) 分隔。

双斜杠并*不*表示路径名；它们仅仅是分隔符，用于分隔保留名称的三个组成部分。如代码示例 4-29 所示，在库目录中，描述保留卷的行以字符 #R 开头。

代码示例 4-29 显示保留卷的库目录

```
      6  00071  00071 lt      0xe8fe    12 9971464 1352412 0x6a000000 131072 0x
#       -il-o-b-----  00-5-24 13:50:02  69-12-31 18:00:00  01-7-13 14:03:00
#R lt 00071 arset0.3// 2001/03/19 18:27:31
     10 ST0001 NO_BAR_CODE lt      0x2741     9 9968052 8537448 0x68000000 1310
#       -il-o-------  07.05.00 15:30:29  69-12-31 18:00:00  13.04.01 13:46:54
#R lt ST0001 hgm1.1// 2001/03/20 17:53:06
     16 SLOT22 NO_BAR_CODE lt      0x76ba     6 9972252 9972252 0x68000000 1310
#       -il-o-------  00-6-6 16:03:05  69-12-31 18:00:00  01-7-12 11:02:05
#R lt SLOT22 arset0.2// 2001/03/02 12:11:25
```

请注意，为符合页宽，代码示例 4-29 中的行已作了删减。

一个或多个保留信息字段可以保留空白，这视 archiver.cmd 文件中定义的选项而定。所列的日期和时间是指创建保留信息的时间。保留行添加到每一个保留卷（即在存档期间保留用于某个存档组的卷）的文件。

可以使用 samu(1M) 实用程序的 v 来显示保留信息，也可使用代码示例 4-30 中所示的 archiver(1M) 或 dump_cat(1M) 命令格式来进行显示。

代码示例 4-30 用于显示保留信息的命令

```
archiver -lv
dump_cat -V catalog_name
```

下面说明了每一种用于显示参数、关键字和分配至卷的保留名示例的格式。

■ 存档组格式。如表 4-17 所示， set 关键字用于激活保留名称中的存档组部分。

表 4-17 存档组格式示例

指令和关键字	保留名称示例
-reserve set	users.1//
	Data.1//

例如，在代码示例 4-31 的 archiver.cmd 文件片段中，以存档组名 allsets 开头的行将按存档组为所有存档组设置保留卷。

代码示例 4-31 按存档组保留卷

```
params
allsets -reserve set
endparams
```

■ 所有者格式。dir、user 和 group 关键字用于激活保留名称中的所有者部分。dir、user 和 group 关键字相互排斥，不能同时使用。dir 关键字使用紧跟在存档组定义的路径参数后面的目录路径部分。user 和 group 是自我说明式的关键字。表 4-18 给出了相关示例。

表 4-18 所有者格式示例

指令和关键字	保留名称示例
-reserve dir	proj.1/p105/
	proj.1/p104/
-reserve user	users.1/user5/
	users.1/user4/
-reserve group	data.1/engineering/

注意 – -reserve 参数用于保留专供一个存档组使用的卷。如果目录很多，但其中的文件较小且数量不多，则会造成很多较小的存档文件写入至每一个保留卷。由于数据文件对每个存档文件的 tar(1) 头文件来说相对较小，因此这些较小的分散文件会降低系统的性能。

■ 文件系统格式。fs 关键字用于激活保留名称中的文件系统部分。表 4-19 给出了相关示例。

表 4-19 文件系统格式示例

指令和关键字	保留名称示例
-reserve fs	proj.1/p103/samfs1
	proj.1/p104/samfs1

第 117 页的 “示例 4”显示了使用保留卷的完整存档示例。

存档程序将卷保留信息记录在库目录文件中。重新标记某个卷后，由于卷中的存档数据实际上已被清除，因此存档程序会自动取消保留该卷。

此外，您还可以使用 reserve(1M) 和 unreserve(1M) 命令分别保留及取消保留卷。有关这些命令的详细信息，请参阅 reserve(1M) 和 unreserve(1M) 手册页。

设置存档优先级：-priority

Sun StorEdge SAM-FS 文件系统为文件的存档操作提供了一个可配置的优先级系统。每一个文件均分配有优先级。文件的优先级是通过文件的属性以及优先级乘数（可在 archiver.cmd 文件中为每一个存档组进行设置）计算出来的。文件属性包括在线 / 离线、存档时限、创建副本的数量和大小。

默认情况下，存档程序不会对存档请求中的文件进行排序，并且所有属性乘数均为零。这将使存档程序按先发现先存档的顺序对文件进行存档。有关优先级的详细信息，请参阅 archiver(1M) 和 archiver.cmd(4) 手册页。

可以通过设置优先级和排序方法，来控制文件的存档顺序。下面是设置优先级的示例：

- 选择 priority 排序方法，以按优先级的顺序对存档请求中的存档文件进行存档。
- 更改 archive_loaded 优先级，以减少介质载入次数。
- 更改 offline 优先级，以使在线文件的存档时间早于离线文件。
- 更改 copy# 优先级，以按副本顺序创建存档副本。

表 4-20 中列出了这些存档优先级。

表 4-20 存档优先级

存档优先级	定义
`-priority age` *value*	存档时限属性乘数
`-priority archive_immediate` *value*	立即存档属性乘数
`-priority archive_overflow` *value*	多个存档卷属性乘数
`-priority archive_loaded` *value*	已载入存档卷属性乘数
`-priority copy1` *value*	副本 1 属性乘数
`-priority copy2` *value*	副本 2 属性乘数
`-priority copy3` *value*	副本 3 属性乘数
`-priority copy4` *value*	副本 4 属性乘数
`-priority copies` *value*	创建副本属性乘数
`-priority offline` *value*	文件离线属性乘数
`-priority queuewait` *value*	队列等待属性乘数
`-priority rearchive` *value*	重新存档属性乘数
`-priority reqrelease` *value*	请求释放属性乘数
`-priority size` *value*	文件大小属性乘数
`-priority stage_loaded` *value*	已载入登台卷属性乘数
`-priority stage_overflow` *value*	多个登台卷属性乘数

其中的 *value*，用于指定一个以下范围之内的浮点数：

```
-3.400000000E+38 ≤ value ≤ 3.402823466E+38
```

安排存档：-startage、 -startcount 和 -startsize

存档程序在扫描文件系统时，会识别其中需要存档的文件。它将那些被识别为存档对象的文件放置在名为一个*存档请求*的列表中。在文件系统扫描结束后，系统会安排对存档请求中的文件进行存档。-startage、 -startcount 和 -startsize 存档组参数控制存档的工作负荷，并确保及时地对文件进行存档。表 4-21 给出了这些参数的格式。

表 4-21 -startage、 -startcount 和 -startsize 指令格式

指令	含义
-startage *time*	指定自扫描过程中标记第一个文件并将其列入存档请求，至开始存档所花费的*时间*。其中的 *time*，用于按第 82 页的 “设置存档时限”中所述的格式指定时间。
-startcount *count*	指定存档请求中可包含的文件数。当存档请求中文件数量达到 *count* 时，开始存档。应为 *count* 指定一个整数。默认情况下，不设置 *count*。
-startsize *size*	指定存档请求中要存档的所有文件的最小总大小 （以字节为单位）。随着存档量逐渐增加，当文件的总大小达到 *size* 时，即开始存档。默认情况下，不设置 *size*。

examine=*method* 指令和 interval=*time* 指令是全局指令，可与 -startage、 -startcount 和 -startsize 指令交互。-startage、 -startcount 和 -startsize 指令可优化存档时间，或者存档完成量。这些值可取代 examine=*method* 参数 （如果已指定）。有关 examine 指令的详细信息，请参阅第 68 页的 “examine 指令：控制存档扫描”。有关 interval 指令的详细信息，请参阅第 69 页的 “interval 指令：指定存档时间间隔”。

在 archiver.cmd 文件中，仅可指定 -startage、 -startcount 和 -startsize 指令中的一个。如果在其中指定了多个指令，那么在满足第一个条件时，即开始存档操作。如果 -startage、 -startcount 和 -startsize 这三者都未指定，则根据 examine=*method* 指令安排存档请求，具体如下：

- 如果 examine=noscan，则当第一个文件进入存档请求中后，存档程序将根据 interval=*time* 指令的参数值来安排存档请求。这属于连续存档法。默认情况下， examine=noscan。
- 如果 examine=scan | scaninodes | scandirs，则存档程序在文件系统扫描完成之后，开始安排存档请求。

archiver.cmd(4) 手册页中给出了如何使用这些指令的示例。

VSN 关联指令

archiver.cmd 文件的 VSN 关联部分用于为存储组分配卷。此部分以 vsns 指令开始，以 endvsns 指令结束。

可以通过下面格式的指令将一组卷分配给存档组：

```
archive_set_name.copy_num  media_type  vsn_expr  ...  [ -pool  vsn_pool_name  ... ]
```

表 4-22　VSN 关联指令的变量

变量	含义
archive_set_name	您的站点为存档组定义的名称。它必须是存档组分配指令中的第一个字段。存档组名通常暗示属于该存档组的文件的特征。指定存档组名时，只能使用 26 个英文字母、数字 (0-9) 和下划线字符 (_)，而不得使用其它特殊字符或空格。存档组名的第一个字符必须是字母。
copy_num	数字后面是一个或多个用于指定该副本存档特征的变量。存档副本指令以数字开头。该数字 （1、 2、 3 或 4）是副本份数。
media_type	介质类型。有关有效介质类型的列表，请参阅 mcf(4) 手册页。
vsn_expr	正则表达式。请参阅 regexp(5) 手册页。
-pool *vsn_pool_name*	已命名的 VSN 组。

一个关联至少需要三个字段：*archive_set_name* 和 *copy_number* 以及 *media_type*，而且至少还需要有一个卷。*archive_set_name* 和 *copy_number* 通过一个句点 (.) 相连。

下面的几个示例以不同的方法指定了相同的 VSN。

示例 1。代码示例 4-32 显示了两行 VSN 参数。

代码示例 4-32　VSN 参数 – 示例 1

```
vsns
set.1  lt  VSN001 VSN002 VSN003 VSN004 VSN005
set.1  lt  VSN006 VSN007 VSN008 VSN009 VSN010
endvsns
```

示例 2。代码示例 4-33 显示了 VSN 参数，它使用反斜杠字符 (\) 将上一行续接至下一行。

代码示例 4-33 VSN 参数 – 示例 2

```
vsns
set.1  lt  VSN001 VSN002 VSN003 VSN004 VSN005 \
 VSN006 VSN007 VSN008 VSN009 VSN010
endvsns
```

示例 3。代码示例 4-34 使用简单的正则表达式指定 VSN。

代码示例 4-34 VSN 参数 – 示例 3

```
vsns
set.1 lt VSN0[1-9] VSN10
endvsns
```

卷由一个或多个 *vsn_expression* 关键字指定，这些关键字即为 `regexp(5)` 手册页中所述的正则表达式。请注意，这些正则表达式所遵守的约定与通配符不同。除正则表达式之外，还可以指定从哪些 VSN 池中选择卷。VSN 池通过 `-pool` *vsn_pool_name* 指令和 VSN 关联来表示。

存档程序需要使用卷对存档组中的文件进行存档时，将检查自动化库和手动载入的驱动器中选定介质类型的每一个卷，以确定它们是否符合 VSN 表达式的要求。存档程序将选择第一个符合表达式要求且包含足够空间以进行存档副本操作的卷。例如：

- 下面的指令指定，将属于存档组 `ex_set` 的第 1 个副本中的文件复制到介质类型 `mo`，且将使用名称为 `optic20` 至 `optic39` 的 20 个卷中的任意卷：

```
ex_set.1 mo optic[2-3][0-9]
```

- 下面的指令将属于存档组 `ex_set` 的第 2 个副本的文件复制到介质类型 `lt`，所使用的卷为以 `TAPE` 开头的任意卷：

```
ex_set.2  lt  ^TAPE
```

如果将 Sun StorEdge SAM-FS 环境配置为按存档组进行回收，则不要将 VSN 分配给多个存档组。

注意 – 在设置 archiver.cmd 文件时，请确保将卷分配给用于存档元数据的存档组。每一个文件系统均有一个与其自身名称相同的存档组。有关保存元数据的详细信息，请参阅 samfsdump(1M) 手册页或 *《Sun QFS、 Sun SAM-FS 和 Sun SAM-QFS 故障恢复指南》*。

VSN 池指令

archiver.cmd 文件中， VSN 池部分以 vsnpools 指令开始，以 endvsnpools 指令结束或至 archiver.cmd 文件的结尾处结束。该部分命名了一组卷。

VSN 池 是一个已命名的卷组。VSN 池非常适用于为某个存档组定义可用的卷，因为它可以提供一个有益于为存档组分配和保留卷的缓冲区。

可以使用 VSN 池定义多个单独的卷组，以供公司中的各部门、组中的各个用户、某些类数据和其它为方便而划分的组使用。系统会为 VSN 池分配名称、介质类型和一组卷。*暂用池* 是指当 VSN 关联中的特定卷或另一个 VSN 池消耗殆尽时，系统临时使用的一组卷。有关 VSN 关联的详细信息，请参阅第 98 页的 “VSN 关联指令”。

如果某个卷被保留用于存档组，则该卷将不能再供它最初所属的 VSN 池使用。因此，已命名的 VSN 池中的卷数量随卷的使用情况而变化。可以按以下格式输入 archiver(1M) 命令来查看 VSN 池：

```
# archiver -lv | more
```

VSN 池定义至少需要三个字段：池名、介质类型以及至少一个 VSN，这些字段以空格分隔。语法格式如下：

```
vsn_pool_name  media_type  vsn_expression
```

表 4-23　VSN 池指令的变量

变量	含义
vsn_pool_name	指定 VSN 池
media_type	由 2 个字符表示的介质类型。有关有效介质类型的列表，请参阅 mcf(4) 手册页。
vsn_expression	正则表达式。可以指定一个或多个 *vsn_expression* 变量。请参阅 regcmp(3G) 手册页。

下面的示例中使用了四个 VSN 池：users_pool、data_pool、proj_pool 和 scratch_pool。如果三个池中某一个池的卷消耗殆尽，那么存档程序选择暂用池 VSN。代码示例 4-35 显示了一个使用四个 VSN 池的 archiver.cmd 文件。

代码示例 4-35 VSN 池示例

```
vsnpools
users_pool   mo ^MO[0-9][0-9]
data_pool    mo ^DA.*
scratch_pool mo ^SC[5-9][0-9]
proj_pool    mo ^PR.*
endvsnpools
vsns
users.1     mo    -pool users_pool   -pool scratch_pool
data.1      mo    -pool data_pool    -pool scratch_pool
proj.1      mo    -pool proj_pool    -pool scratch_pool
endvsns
```

磁盘存档

存档 是指将文件从在线磁盘复制到存档介质的过程。通常，存档程序将存档副本写入至自动化库中磁光盘或磁带卡盒上的卷中；但对于磁盘存档，文件系统中的在线磁盘将用作存档介质。

应用磁盘存档功能，不仅可以将 Sun StorEdge SAM-FS 文件系统中的文件存档至同一计算机主机系统中的另一个文件系统，而且还可将源文件存档至其它 Sun Solaris 系统中的另一文件系统。如果在两个主机系统上应用磁盘存档功能，则其中一个系统为客户机，另一个为服务器。*客户机系统* 是指作为源文件宿主的系统。*服务器系统* 是指作为存档副本宿主的目标系统。

存档文件写入至的文件系统可以是任何一种 UNIX 文件系统，但它不一定必须是 Sun StorEdge SAM-FS 文件系统。如果磁盘存档副本写入至另一台主机，则该主机上必须至少安装了一个 Sun StorEdge SAM-FS 文件系统。

存档程序对待存档至磁盘卷的文件的方式与对待存档至自动化库中卷的文件相同。仍然可以创建一份、两份、三份或四份存档副本。如果创建多份存档副本，则可以将其中一个存档副本写入至磁盘卷，而将其它存档副本写入至可移动介质卷。另外，如果您通常将文件存档至 Sun StorEdge SAM-FS 文件系统中的磁盘卷，则存档程序将根据该文件系统的 archiver.cmd 文件中的规则对存档文件副本自身进行存档。

下面概述了存档至在线磁盘与存档至可移动介质的相似点和不同点：

- 与写入至磁光盘或磁带的存档副本不同，写入至磁盘的存档副本不会记录在目录中。另外，磁盘卷中的存档文件不会出现在历史记录中。
- 如果您要将文件存档至可移动介质卷，则在安装文件系统后，无需更改 `archiver.cmd` 文件中的任何默认值便可开始存档。但是，如果您要将文件存档至磁盘卷，则在安装文件系统之前必须编辑 `archiver.cmd` 文件以定义磁盘存档组。
- 磁盘存档不能使用 `mcf`(4) 文件中的条目。您需要在 `archiver.cmd` 文件中指定 `-disk_archive` 参数，并且需要在 `/etc/opt/ SUNWsamfs/diskvols.conf` 文件中定义磁盘卷。后者是一个附加的配置文件，如果您仅将文件存档至可移动介质卷，则无需使用此文件。

  `diskvols.conf` 文件必须在源文件所在的系统中进行创建。根据存档副本写入位置的不同，此文件还可能包含以下信息：

  - 如果存档副本写入至同一主机系统中的文件系统，则 `diskvols.conf` 文件将定义 VSN 及其路径。
  - 如果存档副本写入至另一个 Sun Solaris 系统，则 `diskvols.conf` 文件将包含该服务器系统的主机名。在此情况下，该服务器系统中也必须有一个 `diskvols.conf` 文件，以定义那些有权向该服务器系统写入数据的客户机。如果希望创建这种客户机 / 服务器关系，在启动第 104 页的 “启用磁盘存档” 过程前，应确保充当服务器的主机上至少安装了一个 Sun StorEdge SAM-FS 文件系统。

配置原则

虽然磁盘存档卷的位置不受限制，但是，建议您不要将该卷设置在源文件所在的磁盘上。如果将客户机系统中的存档副本写入至服务器系统中的磁盘卷，则更加理想。我们建议您创建多个存档副本，并将它们写入至不同类型的存档介质。例如，您可以将第 1 个副本写入至磁盘卷，将第 2 个副本写入至磁带，而将第 3 个副本写入至磁光盘。

如果您将文件存档至服务器上的文件系统，则存档文件自身还会被存档至与目标服务器连接的库中的可移动介质卡盒。

磁盘存档指令

当存档至在线磁盘时，存档程序可识别 `archiver.cmd` 指令中的大多数。它所识别的指令是那些对存档组进行定义以及配置回收过程的指令。而它所忽略的指令是那些在磁盘存档环境中毫无意义的指令，因为这些指令专用于处理可移动介质卡盒。系统将着重识别以下用于磁盘存档组的指令：

- 第 84 页的 “存档组副本参数” 中，除以下指令之外的所有回收指令：

- -fillvsns
- -ovflmin *min_size*
- -reserve *method*
- -tapenonstop

- 第 167 页的 “步骤 2：编辑 archiver.cmd 文件 （可选）”中，除以下指令之外的所有指令：
  - -recycle_dataquantity *size*
  - -recycle_vsncount *count*
- -disk_archive 参数。这是一个存档组处理参数。必须在 archiver.cmd 文件中指定 -disk_archive 参数，以对磁盘存档组进行定义。当数据写入存档磁盘的安装点时，存档程序使用此参数来维护数据的文件系统结构。和所有存档组处理参数相似，它必须位于 params 和 endparams 指令之间。代码示例 4-36 显示了该指令的格式。

代码示例 4-36 -disk_archive 参数的格式

```
params
archive_set.copy_number -disk_archive VSN_Name
endparams
```

其中的 *VSN_Name*，用于指定 diskvols.conf 文件中定义的 VSN。

- clients 和 endclients 指令。如果执行磁盘存档，以将原文件从客户端主机存档至服务器主机，则必须对服务器主机上的 diskvols.conf 文件进行配置。服务器系统上的 diskvols.conf 文件中必须包含客户端系统的名称。这些指令的格式如下所示：

代码示例 4-37 clients 和 endclients 指令的格式

```
clients
client_system1
client_system2
...
endclients
```

其中的 *client_system*，用于指定包含源文件的客户端系统的主机名。

有关磁盘存档指令的详细信息，请参阅 archiver.cmd(4) 手册页。

▼ 启用磁盘存档

磁盘存档功能可随时启用。本节介绍的过程假定存档的前期准备工作已经就绪，现在只需在环境中添加磁盘存档功能。如果您是在初始安装过程中启用磁盘存档，请参阅 *《Sun StorEdge QFS 和 Sun StorEdge SAM-FS 软件安装和配置指南》*以了解相关信息。这时不要使用本过程，因为如果是在安装时添加磁盘存档功能，那么本过程中有些步骤是不必要的。

1. 请确保在要写入磁盘存档副本的主机上，至少安装了一个 Sun StorEdge SAM-FS 文件系统。
2. 以超级用户的身份登录到要存档的文件所在的主机系统。
3. 按照 *《Sun StorEdge QFS 和 Sun StorEdge SAM-FS 软件安装和配置指南》*中所述的步骤启用磁盘存档功能。

   Sun StorEdge SAM-FS 的初始安装过程中包含了一个名为*启用磁盘存档*的步骤。该步骤可以拆分为两个过程。
4. 以超级用户的身份登录到要存档的文件所在的主机。
5. 在包含要存档的文件的主机上，使用 `samd`(1M) `config` 命令应用配置文件的更改，然后重新启动系统。

   例如：

```
# samd config
```

6. 以超级用户的身份登录到将要写入存档副本的主机系统。（可选）

   仅当要将文件存档至另一台主机的磁盘上时，才需要执行此步骤。
7. 在将要写入存档副本的主机上，使用 `samd`(1M) `config` 命令应用配置文件的更改，并重新启动系统。（可选）

   仅当要将文件存档至另一台主机的磁盘上时，才需要执行此步骤。

   例如：

```
# samd config
```

磁盘存档示例

示例 1

代码示例 4-38 显示了客户端系统 pluto 上的 diskvols.conf 文件。

代码示例 4-38 pluto 上的 diskvols.conf 文件

```
# This is file /etc/opt/SUNWsamfs/diskvols.conf on pluto
# VSN Name      [Host Name:]Path
#
disk01                        /sam_arch1
disk02                        /sam_arch2/proj_1
disk03                  mars:/sam_arch3/proj_3
```

在上面的 diskvols.conf 文件中，名为 disk01 和 disk02 的 VSN 将写入至初始源文件所在的主机系统。VSN disk03 将写入至服务器系统 mars 中的 VSN。

代码示例 4-39 显示了服务器系统 mars 上的 diskvols.conf 文件。

代码示例 4-39 mars 上的 diskvols.conf 文件

```
# This is file /etc/opt/SUNWsamfs/diskvols.conf on mars
#
clients
pluto
endclients
```

代码示例 4-40 显示了 pluto 上的 archiver.cmd 文件片断。

代码示例 4-40 pluto 上的 archiver.cmd 文件

```
params
arset1.2 -disk_archive disk01
arset2.2 -disk_archive disk02
arset3.2 -disk_archive disk03
endparams
```

示例 2

在本示例中，文件 /sam1/testdir0/filea 位于存档组 arset0.1 中，存档程序将 /sam1/testdir0/filea 的内容复制到名为 /sam_arch1 的目标路径中。代码示例 4-41 显示了这个 diskvols.conf 文件。

代码示例 4-41 diskvols.conf 文件

```
# This is file /etc/opt/SUNWsamfs/diskvols.conf
#
# VSN Name      [Host Name:]Path
#
disk01                    /sam_arch1
disk02                    /sam_arch12/proj_1
```

代码示例 4-42 显示了 archiver.cmd 文件中与磁盘存档有关的指令行：

代码示例 4-42 archiver.cmd 文件中与磁盘存档有关的指令

```
.
.
.
params
arset0.1 -disk_archive disk01
endparams
.
.
.
```

下面是对已存档至磁盘的 filea 文件运行 sls(1) 命令时的输出。注意代码示例 4-43 中的以下细节：

- dk 是磁盘存档介质的介质类型
- disk01 是 VSN
- f192 是磁盘存档 tar(1) 文件的路径

代码示例 4-43 sls(1M) 的输出

```
# sls -D /sam1/testdir0/filea
/sam1/testdir0/filea:
  mode: -rw-r-----  links:    1  owner: root      group: other
  length:    797904  admin id:       0  inode:      3134.49
  archdone;
  copy 1: ---- Dec 16 14:03        c0.1354 dk disk01 f192
  access:      Dec 19 10:29  modification: Dec 16 13:56
  changed:     Dec 16 13:56  attributes:   Dec 19 10:29
  creation:    Dec 16 13:56  residence:    Dec 19 10:32
```

示例 3

在本示例中，文件 /sam2/my_proj/fileb 位于客户端主机 snickers 的存档组 arset0.1 中，存档程序将该文件的内容复制到服务器主机 mars 的目标路径 /sam_arch1 中。

代码示例 4-44 显示了 snickers 上的 diskvols.conf 文件。

代码示例 4-44 snickers 上的 diskvols.conf 文件

```
# This is file /etc/opt/SUNWsamfs/diskvols.conf on snickers
#
# VSN Name    [Host Name:]Path
#
disk01         mars:/sam_arch1
```

代码示例 4-45 显示了 mars 上的 diskvols.conf 文件。

代码示例 4-45 mars 上的 diskvols.conf 文件

```
# This is file /etc/opt/SUNWsamfs/diskvols.conf on mars
#
clients
snickers
endclients
```

代码示例 4-46 显示了 archiver.cmd 文件中与本示例有关的指令。

代码示例 4-46 archiver.cmd 文件中与磁盘存档有关的指令

```
.
.
.
params
arset0.1 -disk_archive disk01
endparams
.
.
.
```

存档程序示例

表 4-24 显示了本节所有示例使用的目录结构。

表 4-24 目录结构示例

最高层目录	第 1 级子目录	第 2 级子目录	第 3 级子目录
/sam	/projs	/proj_1	/katie
/sam	/projs	/proj_1	/sara
/sam	/projs	/proj_1	/wendy
/sam	/projs	/proj_2	/joe
/sam	/projs	/proj_2	/katie
/sam	/users	/bob	
/sam	/users	/joe	
/sam	/users	/katie	
/sam	/users	/sara	
/sam	/users	/wendy	
/sam	/data		
/sam	/tmp		

示例 1

本示例介绍在无 `archiver.cmd` 文件可用时存档程序的操作。在本示例中， Sun StorEdge SAM-FS 环境包括一个文件系统、一个配有两个驱动器的光盘自动化库、六个卡盒。

代码示例 4-47 显示了 archiver(1M) -lv 命令的输出。它表明存档程序选择的默认介质类型为 mo。且只有 mo 介质可用。

代码示例 4-47 archiver(1M) -lv 输出示例之一

```
# archiver -lv
Notify file: /etc/opt/SUNWsamfs/scripts/archiver.sh

Archive media:
media:lt archmax: 512.0M Volume overflow not selected
media:mo archmax:   4.8M Volume overflow not selected
```

代码示例 4-48 的输出表明存档程序使用了两个驱动器。它列出了 12 个卷及其存储容量和可用空间。

代码示例 4-48 archiver(1M) -lv 输出示例之二

```
Archive libraries:
Device:hp30 drives_available:2 archive_drives:2
  Catalog:
  mo.optic00          capacity:  1.2G space: 939.7M  -il-o-------
  mo.optic01          capacity:  1.2G space: 934.2M  -il-o-------
  mo.optic02          capacity:  1.2G space: 781.7M  -il-o-------
  mo.optic03          capacity:  1.2G space:   1.1G  -il-o-------
  mo.optic10          capacity:  1.2G space:  85.5M  -il-o-------
  mo.optic11          capacity:  1.2G space:  0      -il-o-------
  mo.optic12          capacity:  1.2G space: 618.9k  -il-o-------
  mo.optic13          capacity:  1.2G space: 981.3M  -il-o-------
  mo.optic20          capacity:  1.2G space:   1.1G  -il-o-------
  mo.optic21          capacity:  1.2G space:   1.1G  -il-o-------
  mo.optic22          capacity:  1.2G space: 244.9k  -il-o-------
  mo.optic23          capacity:  1.2G space:   1.1G  -il-o-------
```

代码示例 4-49 的输出表明元数据和数据文件均包括在存档组 samfs 中。当文件的存档时间达到默认的四分钟 （240 秒）时，存档程序将开始创建文件的副本。

代码示例 4-49 archiver(1M) -lv 输出示例之三

```
Archive file selections:
Filesystem samfs  Logfile:
samfs Metadata
    copy:1  arch_age:240
samfs1  path:.
    copy:1  arch_age:240
```

代码示例 4-50 的输出表明存档组中的文件按指定的顺序存档至卷中。

代码示例 4-50 archiver(1M) -lv 输出示例之四

```
Archive sets:
allsets
samfs.1
 media: mo (by default)
 Volumes:
   optic00
   optic01
   optic02
   optic03
   optic10
   optic12
   optic13
   optic20
   optic21
   optic22
   optic23
 Total space available:   8.1G
```

示例 2

本示例说明如何将数据文件和元数据划分至两个不同的存档组。除了第 106 页的“示例 2”中所述的光盘自动化库之外，文件系统环境还配备了手动安装的 DLT 磁带驱动器。较大的文件存档至磁带，而较小的文件存档至光盘卡盒。

代码示例 4-51 显示了 archiver.cmd 文件的内容。

代码示例 4-51 archiver(1M) -lv 输出示例之一：显示 archiver.cmd 文件

```
# archiver -lv -c example2.cmd
Reading archiver command file "example2.cmd"
1: # Example 2 archiver command file
2: # Simple selections based on size
3:
4: logfile = /var/opt/SUNWsamfs/archiver/log
5: interval = 5m
6:
7: # File selections.
8: big . -minsize 500k
9: all .
10:    1 30s
11:
12: vsns
13: samfs.1 mo .*0[0-2]        # Metadata to optic00 - optic02
14: all.1 mo .*0[3-9] .*[1-2][0-9]  # All others for files
15: big.1 lt .*
16: endvsns
```

代码示例 4-52 显示了要使用的介质和驱动器，但没有显示 DLT 及其默认设置。

代码示例 4-52 archiver(1M) -lv 输出示例之二：显示介质和驱动器

```
Notify file: /etc/opt/SUNWsamfs/scripts/archiver.sh
Archive media:
media:lt archmax: 512.0M Volume overflow not selected
media:mo archmax:   4.8M Volume overflow not selected
Archive libraries:
Device:hp30 drives_available:0 archive_drives:0
  Catalog:
  mo.optic00          capacity:  1.2G space: 939.7M  -il-o-------
  mo.optic01          capacity:  1.2G space: 934.2M  -il-o-------
  mo.optic02          capacity:  1.2G space: 781.7M  -il-o-------
  mo.optic03          capacity:  1.2G space:   1.1G  -il-o-------
  mo.optic04          capacity:  1.2G space: 983.2M  -il-o-------
  mo.optic10          capacity:  1.2G space:  85.5M  -il-o-------
  mo.optic11          capacity:  1.2G space:   0     -il-o-------
  mo.optic12          capacity:  1.2G space: 618.9k  -il-o-------
  mo.optic13          capacity:  1.2G space: 981.3M  -il-o-------
  mo.optic20          capacity:  1.2G space:   1.1G  -il-o-------
  mo.optic21          capacity:  1.2G space:   1.1G  -il-o-------
  mo.optic22          capacity:  1.2G space: 244.9k  -il-o-------
  mo.optic23          capacity:  1.2G space:   1.1G  -il-o-------
Device:lt40 drives_available:0 archive_drives:0
```

代码示例 4-52 archiver(1M) -lv 输出示例之二：显示介质和驱动器 *（接上页）*

```
   Catalog:
   lt.TAPE01          capacity:  9.5G space:  8.5G  -il-o-------
   lt.TAPE02          capacity:  9.5G space:  6.2G  -il-o-------
   lt.TAPE03          capacity:  9.5G space:  3.6G  -il-o-------
   lt.TAPE04          capacity:  9.5G space:  8.5G  -il-o-------
   lt.TAPE05          capacity:  9.5G space:  8.5G  -il-o-------
   lt.TAPE06          capacity:  9.5G space:  7.4G  -il-o-------
```

代码示例 4-53 显示了文件系统的组织结构。大于 512000 字节 (500 KB) 的文件在四分钟后存档；其它所有文件在 30 秒后存档。

代码示例 4-53 archiver(1M) -lv 输出示例之三：显示文件系统组织结构

```
Archive file selections:
Filesystem samfs  Logfile: /var/opt/SUNWsamfs/archiver/log
samfs Metadata
    copy:1  arch_age:240
big  path:. minsize:502.0k
    copy:1  arch_age:240
all  path:.
    copy:1  arch_age:30
```

代码示例 4-54 的输出显示了存档组在可移动介质中的分割情况。

代码示例 4-54 archiver(1M) -lv 输出示例之四：显示存档组和可移动介质

```
Archive sets:
allsets
all.1
 media: mo
Volumes:
   optic03
   optic04
   optic10
   optic12
   optic13
   optic20
   optic21
   optic22
   optic23
 Total space available:   6.3G
big.1
 media: lt
Volumes:
   TAPE01
   TAPE02
```

代码示例 4-54 archiver(1M) -lv 输出示例之四：显示存档组和可移动介质 *（接上页）*

```
    TAPE03
    TAPE04
    TAPE05
    TAPE06
  Total space available:  42.8G
samfs.1
 media: mo
Volumes:
    optic00
    optic01
    optic02
  Total space available:   2.6G
```

示例 3

在本示例中，用户文件和项目数据文件存档至不同的介质。data 目录中的文件按大小分别存储至光盘介质和磁带介质。分配至组 ID pict 的文件将被分配至另一个卷组。存档程序不对目录 tmp 和 users/bob 中的文件进行存档。存档程序每隔 15 分钟进行一次存档，并且保存存档记录。

代码示例 4-55 显示了这个示例。

代码示例 4-55 archiver(1M) -lv -c 命令的输出

```
# archiver -lv -c example3.cmd
Reading archiver command file "example3.cmd”
1: # Example 3 archiver command file
2: # Segregation of users and data
3:
4: interval = 30s
5: logfile = /var/opt/SUNWsamfs/archiver/log
6:
7: no_archive tmp
8:
9: fs = samfs
10: no_archive users/bob
11: prod_big data -minsize 50k
12:     1 1m 30d
13:     2 3m
14: prod data
15:     1 1m
16: proj_1 projs/proj_1
17:     1 1m
18:     2 1m
```

代码示例 4-55　archiver(1M) -lv -c 命令的输出 *（接上页）*

```
19: joe . -user joe
20:    1 1m
21:    2 1m
22: pict . -group pict
23:    1 1m
24:    2 1m
25:
26: params
27: prod_big.1 -drives 2
28: prod_big.2 -drives 2
29: endparams
30:
31: vsns
32: samfs.1 mo optic0[0-1]$
33: joe.1 mo optic01$
34: pict.1 mo optic02$
35: pict.2 mo optic03$
36: proj_1.1 mo optic1[0-1]$
37: proj_1.2 mo optic1[2-3]$
38: prod.1 mo optic2.$
39: joe.2 lt 0[1-2]$
40: prod_big.1 lt 0[3-4]$
41: prod_big.2 lt 0[5-6]$
42: endvsns

Notify file: /etc/opt/SUNWsamfs/scripts/archiver.sh

Archive media:
media:lt archmax: 512.0M Volume overflow not selected
media:mo archmax:   4.8M Volume overflow not selected

Archive libraries:
Device:hp30 drives_available:0 archive_drives:0
  Catalog:
  mo.optic00          capacity:  1.2G space: 939.7M  -il-o-------
  mo.optic01          capacity:  1.2G space: 934.2M  -il-o-------
  mo.optic02          capacity:  1.2G space: 781.7M  -il-o-------
  mo.optic03          capacity:  1.2G space:   1.1G  -il-o-------
  mo.optic04          capacity:  1.2G space: 983.2M  -il-o-------
  mo.optic10          capacity:  1.2G space:  85.5M  -il-o-------
  mo.optic11          capacity:  1.2G space:  0      -il-o-------
  mo.optic12          capacity:  1.2G space: 618.9k  -il-o-------
  mo.optic13          capacity:  1.2G space: 981.3M  -il-o-------
  mo.optic20          capacity:  1.2G space:   1.1G  -il-o-------
  mo.optic21          capacity:  1.2G space:   1.1G  -il-o-------
  mo.optic22          capacity:  1.2G space: 244.9k  -il-o-------
  mo.optic23          capacity:  1.2G space:   1.1G  -il-o-------
```

代码示例 4-55 archiver(1M) -lv -c 命令的输出 *（接上页）*

```
Device:lt40 drives_available:0 archive_drives:0
  Catalog:
  lt.TAPE01          capacity:  9.5G space:  8.5G  -il-o-------
  lt.TAPE02          capacity:  9.5G space:  6.2G  -il-o-------
  lt.TAPE03          capacity:  9.5G space:  3.6G  -il-o-------
  lt.TAPE04          capacity:  9.5G space:  8.5G  -il-o-------
  lt.TAPE05          capacity:  9.5G space:  8.5G  -il-o-------
  lt.TAPE06          capacity:  9.5G space:  7.4G  -il-o-------

Archive file selections:
Filesystem samfs  Logfile: /var/opt/SUNWsamfs/archiver/log
samfs Metadata
    copy:1  arch_age:240
no_archive Noarchive path:users/bob
prod_big  path:data minsize:50.2k
    copy:1  arch_age:60 unarch_age:2592000
    copy:2  arch_age:180
prod  path:data
    copy:1  arch_age:60
proj_1  path:projs/proj_1
    copy:1  arch_age:60
    copy:2  arch_age:60
joe  path:. uid:10006
    copy:1  arch_age:60
    copy:2  arch_age:60
pict  path:. gid:8005
    copy:1  arch_age:60
    copy:2  arch_age:60
no_archive Noarchive path:tmp
samfs  path:.
    copy:1  arch_age:240

Archive sets:
allsets

joe.1
 media: mo
 Volumes:
   optic01
 Total space available: 934.2M

joe.2
 media: lt
 Volumes:
   TAPE01
   TAPE02
```

代码示例 4-55　archiver(1M) -lv -c 命令的输出 *（接上页）*

```
 Total space available:  14.7G

pict.1
 media: mo
 Volumes:
   optic02
 Total space available: 781.7M

pict.2
 media: mo
 Volumes:
   optic03
 Total space available:   1.1G

prod.1
 media: mo
 Volumes:
   optic20
   optic21
   optic22
   optic23
 Total space available:   3.3G

prod_big.1
 media: lt drives:2
 Volumes:
   TAPE03
   TAPE04
 Total space available:  12.1G

prod_big.2
 media: lt drives:2
 Volumes:
   TAPE05
   TAPE06
 Total space available:  16.0G

proj_1.1
 media: mo
 Volumes:
   optic10
 Total space available:  85.5M

proj_1.2
 media: mo
 Volumes:
   optic12
```

代码示例 4-55 archiver(1M) -lv -c 命令的输出 *（接上页）*

```
    optic13
  Total space available: 981.9M

samfs.1
  media: mo
  Volumes:
    optic00
    optic01
  Total space available:   1.8G
```

示例 4

在本示例中，用户文件和项目数据文件存档至光盘介质。请注意，代码示例 4-56 没有使用表 4-24 中所示的目录结构。

本示例定义了四个 VSN 池；其中三个池分别用于用户、数据和项目，另一个是暂用池。当 proj_pool 中的介质用尽后，它使用 scratch_pool 来保留卷。本示例介绍如何依据存档组、所有者和文件系统等各个组成部分，来为每一个存档组保留卷。存档程序每隔 10 分钟进行一次存档，并且保存存档日志。

代码示例 4-56 显示了 archiver.cmd 文件及存档程序的输出。

代码示例 4-56 archiver.cmd 文件及存档程序的输出

```
Reading archiver command file "example4.cmd"
1: # Example 4 archiver command file
2: # Using 4 VSN pools
3:
4: interval = 30s
5: logfile = /var/opt/SUNWsamfs/archiver/log
6:
7: fs = samfs
8: users users
9:      1 10m
10:
11: data data
12:     1 10m
13:
14: proj projects
15:     1 10m
16:
17: params
18: users.1 -reserve user
19: data.1 -reserve group
20: proj.1 -reserve dir -reserve fs
```

代码示例 4-56 archiver.cmd 文件及存档程序的输出 *（接上页）*

```
Reading archiver command file "example4.cmd"
21: endparams
22:
23: vsnpools
24: users_pool mo optic0[1-3]$
25: data_pool mo optic1[0-1]$
26: proj_pool mo optic1[2-3]$
27: scratch_pool mo optic2.$
28: endvsnpools
29:
30: vsn
31: samfs.1 mo optic00
32: users.1 mo -pool users_pool -pool scratch_pool
33: data.1 mo -pool data_pool -pool scratch_pool
34: proj.1 mo -pool proj_pool -pool scratch_pool
35: endvsns

Notify file: /etc/opt/SUNWsamfs/scripts/archiver.sh

Archive media:
media:mo archmax:   4.8M Volume overflow not selected

Archive libraries:
Device:hp30 drives_available:0 archive_drives:0
  Catalog:
  mo.optic00         capacity:  1.2G space: 939.7M  -il-o-------
  mo.optic01         capacity:  1.2G space: 934.2M  -il-o-------
  mo.optic02         capacity:  1.2G space: 781.7M  -il-o-------
  mo.optic03         capacity:  1.2G space:   1.1G  -il-o-------
  mo.optic04         capacity:  1.2G space: 983.2M  -il-o-------
  mo.optic10         capacity:  1.2G space:  85.5M  -il-o-------
  mo.optic11         capacity:  1.2G space:   0     -il-o-------
  mo.optic12         capacity:  1.2G space: 618.9k  -il-o-------
  mo.optic13         capacity:  1.2G space: 981.3M  -il-o-------
  mo.optic20         capacity:  1.2G space:   1.1G  -il-o-------
  mo.optic21         capacity:  1.2G space:   1.1G  -il-o-------
  mo.optic22         capacity:  1.2G space: 244.9k  -il-o-------
  mo.optic23         capacity:  1.2G space:   1.1G  -il-o-------

Archive file selections:
Filesystem samfs  Logfile: /var/opt/SUNWsamfs/archiver/log
samfs Metadata
    copy:1  arch_age:240
users  path:users
    copy:1  arch_age:600
data  path:data
    copy:1  arch_age:600
```

代码示例 4-56 archiver.cmd 文件及存档程序的输出 *（接上页）*

```
Reading archiver command file "example4.cmd"
proj  path:projects
     copy:1  arch_age:600
samfs  path:.
     copy:1  arch_age:240


VSN pools:
data_pool media: mo Volumes:
   optic10
 Total space available:  85.5M

proj_pool media: mo Volumes:
   optic12
   optic13
 Total space available: 981.9M

scratch_pool media: mo Volumes:
   optic20
   optic21
   optic22
   optic23
 Total space available:   3.3G

users_pool media: mo Volumes:
   optic01
   optic02
   optic03
 Total space available:   2.7G

Archive sets:
allsets

data.1
  reserve:/group/
 media: mo
 Volumes:
   optic10
   optic20
   optic21
   optic22
   optic23
 Total space available:   3.4G

proj.1
  reserve:/dir/fs
 media: mo
```

代码示例 4-56 archiver.cmd 文件及存档程序的输出 *（接上页）*

```
Reading archiver command file "example4.cmd"
 Volumes:
   optic12
   optic13
   optic20
   optic21
   optic22
   optic23
 Total space available:   4.2G

samfs.1
 media: mo
 Volumes:
   optic00
 Total space available: 939.7M

users.1
  reserve:/user/
 media: mo
 Volumes:
   optic01
   optic02
   optic03
   optic20
   optic21
   optic22
   optic23
 Total space available:   6.0G
```

存档程序原则

存档程序使用 archiver.cmd 文件进行自动存储管理操作。编写此文件之前，请复习一些可以改善 Sun StorEdge SAM-FS 文件系统和存档程序性能的通用原则，这是非常有益的，可确保您以最安全的方式存储数据。

每一个站点在计算应用、数据存储硬件和软件方面都是各不相同的。下面的建议是 Sun Microsystems 根据多年的经验总结出来的。为您的站点编写 archiver.cmd 文件时，请确保通过考虑以下方面来反映站点的数据存储要求。

1. 保存存档日志。存档日志可为数据恢复提供非常重要的信息，即便在 Sun StorEdge SAM-FS 软件无法使用时也是如此。我们建议您将这些日志保存在安全的地方，以防因发生灾难性故障而造成 Sun StorEdge SAM-FS 软件无法使用。

2. 使用正则表达式指定卷。允许系统将文件存储在多个不同的卷上。通过正则表达式指定的卷范围可以使系统连续不断地运行。如果您为存档组副本指定特定的卷名，则会造成数据很快充满卷。因此，您不得不频繁地更换介质，从而导致不应有的工作流程问题。

3. 根据文件的创建和修改频率以及是否需要保存所有修改副本，来设置您的存档时间间隔。请注意，存档时间间隔是指对文件系统执行扫描操作的时间间隔。如果将存档时间间隔设置得太短，则会使存档程序几乎不间断地执行扫描。

4. 考虑您要使用的文件系统数量。与单个 Sun StorEdge SAM-FS 文件系统相比，多个 Sun StorEdge SAM-FS 文件系统通常可以提高存档程序的性能。存档程序为每一个文件系统运行单独的进程。多个文件系统的扫描时间要比单个文件系统少得多。

5. 与在 UNIX 文件系统中一样， Sun StorEdge SAM-FS 文件系统中的文件也可使用目录结构来组织。出于性能考虑， Sun Microsystems 建议您不要将 10,000 个以上的文件放入同一个目录中。

6. 请始终制作两份文件副本，并将它们存储至不同的卷上。如果将数据存储至同一种介质类型，则在介质出现物理问题时，您会面临数据丢失的风险。如果条件允许，请制作多份存档副本。

7. 务必定期运行 `samfsdump`(1M) 来转储您的元数据。元数据 （包括目录结构和文件名等）存储在与文件系统同名的存档组中。在出现故障时，您可以使用此类信息来恢复文件系统。如果您不想执行此操作，可以通过将此存档组分配至不存在的 VSN，来防止存档此类数据。有关保存元数据的详细信息，请参阅 *《Sun QFS、 Sun SAM-FS 和 Sun SAM-QFS 故障恢复指南》*或 *《Sun StorEdge QFS 和 Sun StorEdge SAM-FS 软件安装和配置指南》*。

排除存档程序故障

存档程序完成安装并初次运行的时候，执行任务的结果可能不尽如人意。请确保使用下列工具来监视系统的存档活动：

- `samu`(1M) 实用程序的 `a` 显示选项。此显示命令可以显示每一个文件系统的存档程序活动。另外，它还可以显示如下所示错误和警告消息：

```
存档程序指令中存在错误 – 将不会存档。
```

  `samu`(1M) 实用程序的 `a` 显示屏幕中包括了每个文件系统的有关消息。它指明存档程序何时将再次扫描 `.inodes` 文件，以及目前正在对哪些文件进行存档。

- 存档日志。可以在 archiver.cmd 文件中定义这些日志，并且应该定期检查这些日志，以确保文件已经存档至卷中。存档日志可能会变得非常大，您应定期手动或使用 cron(1) 作业来减小它。为安全起见，请对这些日志文件进行存档，因为这些信息可用于恢复数据。
- sfind(1)。使用此命令定期检查未被存档的文件。如果发现未被存档的文件，请确保了解它们未被存档的原因。
- sls(1)。除非文件存在有效的存档副本，否则系统不会释放该文件占用的磁盘空间。使用 sls -D 命令可以显示文件的索引节点信息 （包括副本信息）。

注意 – 对于某个文件，sls -D 命令的输出可能会显示 archdone 字样。这并不表示文件已具有存档副本。它只是表示存档程序已扫描该文件，并且已完成所有与其自身相关的工作。只有通过查看 sls(1) 命令显示的副本信息，才能确定文件是否具有存档副本。

有时，您可能会看到一些表明存档程序已用尽卡盒空间或卡盒不存在的消息，这些消息如下所示：

- 当存档程序未发现分配给存档组的卡盒时，会显示以下消息：

```
无可用于存储存档组 setname 的卷
```

- 当存档程序发现分配给存档组的卡盒无可用空间时，会显示以下消息：

```
No space available on Archive Set setname
```

文件未被存档的原因

以下列表显示了您的 Sun StorEdge SAM-FS 环境没有对文件进行存档的原因。

1. archiver.cmd 文件存在语法错误。运行 archiver -lv 命令查找错误，然后更正标有错误标记的行。
2. archiver.cmd 文件中存在 wait 指令。删除此 wait 指令或运行 samu(1M) 实用程序的 :arrun 命令来取代该指令。
3. 无可用的卷。可以查看 archiver(1M) v 命令的输出，来确定是否存在此情况。根据需要添加更多的卷。您可能需要导出现有的卡盒以腾出自动化库中的端口。
4. 存档组的卷已满。您可以导出卡盒并导入新的替换卡盒 （确保已标记新卡盒），也可对卡盒执行回收操作。有关回收的详细信息，请参阅第 159 页的 “回收”。

5. archiver.cmd 文件中的 VSN 部分列出的介质不正确。检查正则表达式和 VSN 池，确保正确定义了它们。

6. 可用卷上无足够的空间来存档任何文件。如果您的文件较大，并且卷似乎已满，则卡盒的利用率可能已达到 Sun StorEdge SAM-FS 环境规定的界限。如果出现这种情况，请添加新卡盒或执行回收操作。

   如果您已指定 -join path 参数，并且无足够的空间将目录中的所有文件存档至任何卷，则不会进行存档。这时，您应该添加新的卡盒，进行回收，或使用以下某一参数：-sort path 或 -rsort path。有关这些参数的详细信息，请参阅第 89 页的 “联合存档：-join”。

7. archiver.cmd 文件为包含大型文件的目录或文件系统设置了 no_archive 指令。

8. 使用 archive(1) -n （永不存档）命令设置了太多的目录，导致其中的文件永远无法存档。

9. 大型文件使用频繁。因此，它们永远达不到存档时限而无法存档。

10. 自动化库存在硬件问题或配置问题。

11. 客户端与服务器之间的网络连接有问题。确保客户端和服务器之间已建立稳定的通信。

其它存档程序诊断方法

在排除存档程序的故障时，除检查上表列出的项目之外，还应检查以下项目。

1. syslog 文件 （默认位于 /var/adm/sam-log 目录下）。此文件包含的存档程序消息可以指出问题的根源。

2. 卷容量。确保所有必需的卷可用，并且它们具有足够的存档空间。

3. 如果存档程序产生过多的原因不明的卡盒活动，或似乎未执行任何活动，请打开跟踪工具并检查跟踪文件。有关跟踪文件的详细信息，请参阅 defaults.conf(4) 手册页。

4. 您可以对存档程序进程 (sam-archiverd) 运行 truss(1) -p *pid* 命令，以确定无响应的系统调用。有关 truss(1) 命令的详细信息，请参阅 truss(1) 手册页。

5. showqueue(1M) 命令可以显示存档程序队列文件的内容。您可以使用此命令来检查正在安排的或已存档的存档请求的状态。任何无法安排的存档请求均会生成一则说明原因的消息。另外，此命令还可以显示存档进度。

文件未被释放的原因

存档程序和释放程序相互合作来协调磁盘高速缓存中的可用数据量。未从磁盘高速缓存自动释放文件的主要原因是它们尚未存档。

有关文件未被释放的原因的详细信息，请参阅第 140 页的 “排除释放程序的故障”。

第五章

释放

释放 是指释放程序通过识别已存档的文件并释放这些文件在磁盘高速缓存中的副本，从而使磁盘高速缓存空间可再利用的过程。这可以为其它从存档介质中创建或登台的文件腾出空间。释放程序只释放已存档的文件。释放文件后，系统会从磁盘高速缓存中清除该文件的所有数据。

当高速缓存的占用量达到站点指定的磁盘阀值时， Sun StorEdge SAM-FS 文件系统会自动调用释放进程。与释放程序不同， `release`(1) 命令可以立即释放文件占用的磁盘空间或为文件设置释放参数。有关释放进程的详细信息，请参阅 `sam-releaser`(1M) 手册页。

释放程序所包含的功能允许您指定哪些文件在存档之后立即释放、哪些文件永不释放以及哪些文件可以部分释放。由于某些诸如 `filemgr`(1) 之类的实用程序只读取文件的起始部分，因此部分释放功能特别有用。采用部分释放功能时，文件的一部分保留在磁盘高速缓存中，而文件的剩余部分会被释放。读取仍保留在磁盘高速缓存中的文件的第一部分时，并不会导致系统从存档介质中将文件的剩余部分重新登台至磁盘高速缓存。本章即介绍上述功能及其它多种功能。

本章包括下列主题：

释放程序概述

当文件系统的利用率超过所配置的上限时，文件系统管理软件将启动释放程序。首先，释放程序读取 `releaser.cmd` 文件并收集用于控制释放进程的指令。其次，它扫描文件系统并收集每一个文件的有关信息。最后，在扫描整个文件系统后，释放程序开始按优先级顺序释放文件。

只要文件系统的利用率高于所配置的下限，释放程序就会继续释放文件。通常，释放程序会释放足够的空间，以使文件系统的利用率低于所配置的下限。如果释放程序没有发现任何需要释放的文件，将会退出。以后，当更多的文件可以释放时，释放程序即会运行。当高于上限时，文件系统会每隔一分钟启动一次释放程序。

上、下限可以使用 `high=`*percent* 和 `low=`*percent* 文件系统安装选项来设置。有关安装选项的详细信息，请参阅 `mount_samfs`(1M) 手册页。

操作原理

文件系统可能包含成千上万个文件。由于只需释放几个大文件便有可能使文件系统的利用率降至下限，因此没有必要记录所有文件的释放优先级。但是，释放程序又必须检查每一个文件的优先级，否则就不能释放最恰当的备选文件。释放程序通过只确定前 10,000 个备选文件来解决这一问题。

确定前 10,000 个备选文件之后，如果随后的备选文件优先级不高于前 10,000 个备选文件的最低优先级，则释放程序会忽略随后的备选文件。

确定前 10,000 个备选文件的优先级之后，释放程序会选择释放具有最高优先级的文件。每释放一个文件，释放程序就会进行一次检查，确定文件系统的高速缓存利用率是否低于下限。如果是，释放程序将停止释放文件。如果否，释放程序将继续按优先级的顺序释放文件。

如果在释放全部 10,000 个备选文件之后，文件系统的利用率仍高于下限，释放程序将重新确定 10,000 个新备选文件。

如果找不到任何合适的备选文件，释放程序会退出。例如，在文件没有存档副本时，就会出现这种情况。一分钟后， Sun StorEdge SAM-FS 文件系统将再次启动释放程序。

定义

本节介绍本章中使用的术语。

时限

时限 是指从发生指定事件开始到现在所经历的时间。文件的索引节点可以跟踪记录释放程序所使用的下列时间：

- 驻留更改时间
- 数据修改时间
- 数据访问时间

您可以使用带有 `-D` 选项的 `sls`(1) 命令来查看这些时间。每一种时间都有对应的时限。例如，如果当前时间是上午 10 点 15 分，则在上午 10 点 10 分所修改的文件的数据修改时限为 5 分钟。有关 `sls`(1) 命令的详细信息，请参阅 `sls`(1) 手册页。

备选文件

备选文件 即符合释放条件的文件。文件不能成为备选文件的原因包括：

- 该文件已离线。
- 该文件尚未存档。
- `archiver.cmd` 命令文件为该文件指定了 `-norelease` 属性，并且尚未为该文件创建完所需的副本份数。
- 该文件标记为 “已损坏”。
- 该文件不是普通文件，而是目录文件、块文件、特殊字符文件或管道文件。
- 存档程序正在登台该文件以创建另一副本。登台之后，该文件便成为适合释放的文件。
- 该文件的时限为负数。这种情况通常会发生在未正确设置时钟的 NFS 客户端上。
- 文件被标记为永不释放。这可使用 `release`(1) `-n` 命令来指定。
- 该文件在过去登台的时间小于最短驻留时间设置。有关的详细信息，请参阅第 135 页的 “指定最短驻留时间：min_residence_age”。
- 已用 `release`(1) 命令的 选项将该文件标记为 “部分释放”，并且释放程序已部分释放该文件。
- 该文件太小。

优先级

优先级 是一个表示备选文件级别的数值，该数值取决于用户提供的应用于该备选文件数值属性的权数。总优先级是以下两类优先级之和：时限优先级和大小优先级。

释放程序首先释放具有较大优先级数值的备选文件，然后释放具有较小优先级数值的备选文件。

权数

权数 是一个数值，用于使优先级的计算倾向于包括您感兴趣的文件属性，并排除不感兴趣的文件属性。例如，如果将文件的大小权数设置为零，则在计算优先级时，不会考虑文件的大小属性。权数是介于 0.0 和 1.0 之间的浮点值。

部分释放

可以*部分释放* 文件，方法是指定在磁盘高速缓存中保留文件的起始部分，而释放文件的剩余部分。例如，在使用 `filemgr`(1) 等实用程序读取文件的起始部分时，部分释放功能非常有用。

部分释放和部分登台

释放和登台是两个互为补充的进程。文件在存档后，就可以从在线磁盘高速缓存中完全释放，站点也可以指定只在磁盘高速缓存中保留文件的起始部分 （即*存根* ），而释放文件的其余部分。这种部分释放文件的功能可以使系统在不登台文件的情况下，立即访问文件存根中的数据。

系统管理员可以在安装文件系统时，指定部分释放的默认大小和保持在线的存根的最大值。系统管理员可以在 `mount`(1M) 命令中设置这些大小，具体如下：

- 指定 `-o partial=`*n* 选项，来设置保持在线的文件存根的默认大小 (*n*)。`-o partial=`*n* 的设定值必须小于或等于 `-o maxpartial=`*n* 的设定值。最小设定值为 `-o partial=8` KB。默认设置为 `-o partial=16` KB。
- 指定 `-o maxpartial=`*n* 选项，来设置保持在线的文件存根大小的最大值 (*n*)。要限制可以保持在线的文件存根的大小，可使用 `-o maxpartial=`*n* 选项，并将其值指定为可以保持在线的最大存根大小。要禁用部分释放功能，可指定 `-o maxpartial=0`。

用户可以通过在 release(1) 命令中指定 -p 选项或在 sam_release(3) 库例程中指定 p 选项，来指定文件的默认存根大小。要为不同类型的文件或不同的应用程序指定不同大小的文件存根，用户可以在 release(1) 命令中指定 -s 选项，或在 sam_release(3) 库例程中指定 s 选项。 -s 和 s 的值必须小于安装文件系统时使用 mount(1M) 命令的 -o maxpartial 选项指定的值。

系统管理员可以使用另一个安装选项，即 -o partial_stage=*n*，来确定在登台文件的剩余部分之前，应读取多少部分释放存根。也就是说，当读取的存根量大于 -o partial_stage=*n* 后，即开始登台文件。

默认情况下，系统将 -o partial_stage=*n* 选项设置为等于部分释放存根的大小。虽然用户可以配置该值，但应注意该值对文件登台的影响，具体如下：

- 如果将 -o partial_stage=*n* 选项设置为等于部分释放存根的大小，则系统的默认操作是直到应用程序到达部分释放存根的末尾时才允许登台文件。等待到达存根的末尾会推迟应用程序对文件剩余部分的访问。
- 如果将 -o partial_stage=*n* 选项设置为小于部分释放存根的值，则会出现以下情况。在应用程序超过 -o partial_stage=*n* 选项设置的阀值后，系统将登台文件的剩余部分。这可以加快应用程序对文件数据剩余部分的访问。

示例。假定您设置了下列选项：

- -o partial_stage=16 （即 16 KB）
- -o partial=2097152 （即 2 GB）
- -o maxpartial=2097152 （即 2 GB）

所使用的程序为 filemgr(1)，它首先读取文件的前 8 KB。此时，系统不会登台文件。有一个视频点播程序读取同一个文件，并且当它读完文件的前 16 KB 时，系统开始登台文件。在安装并定位存档磁带后，该应用程序会继续读取 2 GB 的磁盘数据。当视频点播程序读完 2 GB 的文件数据后，它会在登台活动之后立即进行读取。由于在该应用程序读取部分文件数据时已经安装并定位了磁带，因此它不必等待。

有多个命令行选项可影响文件是否可以标记为部分释放。某些选项可由系统管理员启用，而另一些选项可由个别用户启用。以下几节介绍了不同类型的用户可以设置的释放特征。

系统管理员选项概述

系统管理员可以在安装文件系统时，更改部分释放的最大值和默认值。表 5-1 列出了可影响部分释放的 mount(1M) 选项。有关 mount(1) 命令的详细信息，请参阅 mount_samfs(1M) 手册页。

表 5-1 影响部分释放的安装选项

mount(1M) 选项	作用
-o maxpartial=*n*	指定在文件标记为部分释放时，可以在在线磁盘高速缓存中保留的最大空间 (KB)。最大值为 2,097,152 KB，即 2 GB。最小值为 0，即不允许部分释放任何文件。 如果指定 -o maxpartial=0，则会禁用部分释放功能。已释放的文件会被完全释放，并且磁盘高速缓存中不会保留文件的任何部分。一旦安装文件系统，用户便不能再改写此选项指定的值。 默认情况下， *n* 变量设置为 16。此设置使得用户可以将文件标记为部分释放，且这个文件最多可在磁盘上保留 16 KB 的数据。
-o partial=*n*	当用户使用 release(1) 命令的 -p 选项将文件标记为部分释放时，该选项用于设置要在磁盘高速缓存中保留的默认空间 (KB)。 *n* 变量的值至少为 8，但它也可以等于 -o maxpartial=*n* 选项的设定值。 由于某些应用程序不必访问整个文件便可完成其工作，因此该选项可用于确保应用程序能够从文件的起始部分获得所需的信息。同时，使用此选项还可阻止系统登台不必要的文件。 默认值为 -o partial=16。
-o partial_stage=*n*	指定在访问部分释放的文件时，从存档介质登台整个文件之前应读取 *n* 字节的文件数据。此选项的设定值通常小于 -o partial 的设定值。其中的 *n*，用于指定介于 0 和 -o maxpartial 参数之间的一个整数值。默认情况下，系统将它设置为 16 或是由 -o partial 选项指定的任何值。
-o stage_n_window=*n*	将一次可以登台的数据量指定为 *n*。其中的 *n*，用于指定一个介于 64 至 2,048,000 之间的整数。默认值为 256 KB。此选项仅适用于已设置 stage -n 属性的文件。

用户选项概述

系统管理员可以设置文件在释放后，可保留在磁盘高速缓存中的文件存根大小的最大值和默认值。此外，系统管理员还可以确定是否为特定的文件系统启用部分释放功能。

不过，通过使用 release(1) 命令和 sam_release(3) 库例程，用户可以设置其它释放属性以及指定要标记为部分释放的文件。表 5-2 中列出了可以指定部分释放属性的命令和库选项。有关 release(1) 命令的详细信息，请参阅 release(1) 手册页。有关 sam_release(3) 库例程的详细信息，请参阅 sam_release(3) 手册页。

表 5-2 用户释放选项

选项	作用
release(1) 命令和 -p 选项 或 sam_release(3) 库例程和 p 选项	-p 和 p 选项用于将指定的文件标记为部分释放。如果使用这些选项，则文件在释放后可以保留在在线磁盘高速缓存中的数据量，取决于在安装该文件所在的文件系统时为 -o partial=*n* 选项设置的值。这些选项不能用于指定保持在线的字节数。
release(1) 命令和 -s partial_size 选项 或 sam_release(3) 库例程和 s 选项	-s 和 s 选项用于将指定的文件标记为部分释放，并指定要在在线磁盘高速缓存中保留的文件数据量。 -s 或 s 选项的变量用于指定要保持在线的数据量 (KB)。 用户指定的保持在线的文件数据量不能大于安装文件系统时指定的 -o maxpartial=*n* 值。如果用户指定的值大于文件系统的值，则系统会使用文件系统的值，而忽略用户指定的参数值。

releaser.cmd 文件

/etc/opt/SUNWsamfs/releaser.cmd 文件由指定站点特定释放操作的指令行组成。 releaser.cmd 文件可以包含用于设置释放优先级的指令、用于指定日志文件的指令以及用于执行其它操作的指令。

以下几节介绍了 releaser.cmd 文件中的指令：

- 第 132 页的 “指定与文件时限和与文件大小有关的释放优先级指令：weight_age、 weight_age_access、 weight_age_modification 和 weight_age_residence”
- 第 134 页的 “指定用于单个文件系统的指令：fs”
- 第 135 页的 “指定调试指令：no_release 和 display_all_candidates”
- 第 135 页的 “指定最短驻留时间：min_residence_age”
- 第 135 页的 “指定日志文件：logfile”
- 第 137 页的 “阻止释放重新存档的文件：rearch_no_release”
- 第 138 页的 “调整释放程序备选文件列表的大小：list_size”

有关这些指令的详细信息，请参阅 `releaser.cmd`(4) 手册页。

指定与文件时限和与文件大小有关的释放优先级指令：`weight_age`、`weight_age_access`、`weight_age_modification` 和 `weight_age_residence`

释放程序根据 `releaser.cmd` 文件中定义的指令所确定的优先级顺序来释放文件系统中的文件。在释放文件时，它既考虑文件的时限，又考虑文件的大小。默认情况下，站点首先释放最大且最旧的文件，而将最小且最新的文件保留在磁盘中。以下几节介绍了释放程序在确定文件系统中文件的释放优先级时，是如何考虑文件的时限和大小的。

有关这些释放指令的详细信息，请参阅 `releaser.cmd`(4) 手册页。

文件时限

释放程序在确定与时限相关的文件释放优先级要素时，将考虑下列可能的时限：

- 从最后一次访问文件到现在的时限
- 从最后一次修改文件到现在的时限
- 从文件在磁盘高速缓存中的驻留状态发生更改到现在的时限

在某些情况下，您可能希望文件的访问时限优先于修改时限。而在其它情况下，可能优先考虑由最近访问时间、修改时间和驻留状态更改时间得出的简单时限。

默认情况下，文件时限是指下列三个文件时限中最小的一个：

- 文件访问时限
- 文件修改时限
- 文件驻留时限

您可以使用指令指定在计算文件的释放优先级时要使用的加权时限优先级。

代码示例 5-1 显示了时限优先级指令的格式。

代码示例 5-1 时限优先级指令的格式

```
weight_age = float
weight_age_access = float
weight_age_modification = float
weight_age_residence = float
```

- `weight_age` 指令指定要为其分配加权因子的文件默认时限 （文件访问时限、修改时限和驻留时限三者中的较小者）。其中的 *float*，用于指定处于以下范围内的浮点数：0.0 ≤ *float* ≤ 1.0。默认情况下，*float* = 1.0。

  该指令不能与 `weight_age_residence`、 `weight_age_modify` 或 `weight_age_access` 指令结合使用。

- `weight_age_residence`、`weight_age_modify` 和 `weight_age_access` 指令指定由这些可能时限中的一个、二个或三个组合起来而确定的文件时限。其中的 *float*，用于指定处于以下范围内的浮点数：0.0 ≤ *float* ≤ 1.0。默认情况下，*float* = 1.0。

  这些指令不能与 `weight_age` 指令结合使用。

  如果使用 `weight_age_residence`、 `weight_age_modify` 和 `weight_age_access` 指令，则依据这三个时限的组合来计算文件的与时限相关的优先级。首先，收集每一个文件的可能时限数据。其次，将文件时限数据与 `releaser.cmd` 文件中指定的加权因子相乘。最后，将时限数据乘以每一个加权因子的结果相加 （如代码示例 5-2 所示），从而计算出与文件时限相关的优先级：

代码示例 5-2 优先级计算

```
   file access age * weight_age_access
+ file modification age * weight_age_modification
+ file residency age * weight_age_residence

_____________________________________________________
= age_related_priority
```

示例。代码示例 5-3 显示了 `releaser.cmd` 文件中的指令行，它们指定在计算文件的释放优先级时，只考虑文件的驻留时限 （而忽略修改时限和访问时限）。

代码示例 5-3 `releaser.cmd` 文件片断

```
weight_age_residence = 1.0
weight_age_modify = .0
weight_age_access = .0
```

计算出与文件时限相关的优先级之后，将它乘以与文件大小相关的优先级。下一节介绍了如何计算与文件大小相关的优先级。

文件大小

释放程序在确定与大小相关的文件释放优先级要素时将考虑文件的大小。文件的大小 （在 4 KB 块中）乘以您为 `weight_size` 指令指定的权数，即可得出与大小相关的文件释放优先级要素。

weight_size 指令的格式如下所示：

```
weight_size = float
```

其中的 *float*，用于指定处于以下范围内的浮点数：0.0 ≤ *float* ≤ 1.0。默认情况下，*float* = 1.0。

示例。代码示例 5-4 显示了 releaser.cmd 文件，其中指定在计算文件的释放优先级时，忽略 samfs1 和 samfs2 文件系统中所有文件的大小。

代码示例 5-4 releaser.cmd 文件

```
# releaser.cmd file
logfile = /var/adm/default.releaser.log
weight_size = 0.0
#
fs = samfs1
weight_age = 1.0
logfile = /var/adm/samfs1.releaser.log
#
fs = samfs2
weight_age_modify = 0.3
weight_age_access = 0.03
weight_age_residence = 1.0
logfile = /var/adm/samfs1.releaser.log
```

指定用于单个文件系统的指令：fs

可以在 releaser.cmd 文件中使用 fs = *family_set_name* 指令来指定， fs = 之后的指令仅适用于指定的文件系统。此指令的格式如下：

```
fs = family_set_name
```

其中的 *family_set_name*，用于指定 mcf 文件中的系列集名称。

位于第一个 fs = 指令前面的指令是应用于所有文件的全局指令。 fs = 指令后面的指令可以取代全局指令。本章所述的指令既可用作全局指令，也可用作专用于一个文件系统的指令。

releaser.cmd(4) 手册页中提供了 fs = 指令的示例。

指定调试指令：no_release 和 display_all_candidates

在调节或调试释放程序时， no_release 和 display_all_candidates 指令非常有用。这几个指令如下：

- no_release 指令可以防止文件从在线磁盘高速缓存中被删除。可以使用此指令来检查 releaser.cmd 文件中的指令，这不会真正地释放文件。此指令的格式如下：

```
no_release
```

- display_all_candidates 指令可以将所有释放备选文件的名称写入至日志文件。此指令的格式如下：

```
display_all_candidates
```

由于释放程序可将释放备选文件的名称写入至日志文件，而不是真正从文件系统中释放这些文件，因此这些指令对调试很有帮助。

指定最短驻留时间：min_residence_age

min_residence_age 指令可用于指定文件在成为释放备选文件之前，必须在文件系统中驻留的最短时间。此指令的格式如下：

```
min_residence_age = time
```

其中的 *time*，用于指定驻留时间，以秒为单位。默认时间为 600 秒，即 10 分钟。*time* 设置的最大或最小值并没有具体规定。

指定日志文件：logfile

如果在 releaser.cmd 文件中指定了 logfile 指令，释放程序会将其自身活动添加至指定的文件名，或新创建的文件名 （如果指定的文件名不存在）。此指令的格式如下：

```
logfile = filename
```

其中的 *filename*，用于指定日志文件的名称。

代码示例 5-5 是一个日志文件范例 （请注意，为适应页宽已将某些行换行）。

代码示例 5-5 释放程序日志文件示例

```
Releaser begins at Wed Apr 28 17:29:06 1999
inode pathname          /sam1/.inodes
low-water mark          24%
weight_size             1
weight_age              1
fs equipment ordinal    1
family-set name         samfs1
started by sam-amld?    yes
release files?          yes
display_all_candidates? no
---before scan---
blocks_now_free:    3481504
lwm_blocks:         3729362
---scanning---
10501 (R: Wed Apr 21 18:47:50 CDT 1999) 10001 min, 500 blks /sam1/testdir0/filevp
10500 (R: Wed Apr 21 18:48:10 CDT 1999) 10000 min, 500 blks /sam1/testdir0/filewq
...
---after scan---
blocks_now_free:      3730736
lwm_blocks:           3729362
archnodrop: 0
already_offline: 0
bad_inode_number: 0
damaged: 0
extension_inode: 0
negative_age: 0
nodrop: 1
not_regular: 9
number_in_list: 675
released_files: 202
too_new_residence_time: 0
too_small: 2
total_candidates: 675
total_inodes: 1376
wrong_inode_number: 0
zero_arch_status: 689
zero_inode_number: 0
zero_mode: 0
CPU time: 2 seconds.
Elapsed time: 10 seconds.
Releaser ends at Wed Apr 28 17:29:16 1999
```

releaser(1M) 手册页介绍了日志文件中包含的信息。随着释放程序的每一次运行，日志文件的大小会不断增加，因此，请务必注意减小日志文件的大小，或删除 logfile 关键字。

代码示例 5-6 显示了 ---after scan--- 行下，各统计项之间的数学关系：

代码示例 5-6 代码示例 5-5 中 ---after scan--- 行之后各统计项的数学关系

```
total_inodes = wrong_inode_number +
zero_inode_number +
zero_mode +
not_regular +
extension_inode +
zero_arch_status +
already_offline +
damaged +
nodrop +
archnodrop +
too_new_residence_time +
too_small +
negative_age +
total_candidates
   released_files = total_candidates
```

阻止释放重新存档的文件：rearch_no_release

默认情况下，系统会释放标记为重新存档的文件。如果在 releaser.cmd(4) 文件中指定 rearch_no_release 指令，释放程序不会释放标记为重新存档的文件。此指令的格式如下：

```
rearch_no_release
```

调整释放程序备选文件列表的大小：list_size

可使用 list_size 指令来指定释放程序备选文件的数量。如果您注意到释放程序为了达到下限而释放所需数量的文件之前，在对多个文件系统执行扫描，那么可能需要考虑增大这个值以使之大于默认值 10,000。在包含许多小文件的文件系统中就可能存在这样的情况。您可以从释放程序日志文件中，获取有关释放程序的活动信息。此指令的格式如下：

```
list_size = number
```

其中的 number，用于指定范围如下的一个整数：10 ≤ *number* ≤ 2,147,483,648。

archiver.cmd 文件在释放过程中的作用

archiver.cmd 文件中的大多数指令会影响存档过程，但存档组分配指令可以使您指定应用于存档组中所有文件的释放属性。

存档组分配指令的格式如下：

```
archive_set_name  path  [search_criteria ...]  directives  ...
```

表 5-3 显示了与释放有关的 *directives （指令）*。

表 5-3 存档组分配 *directives （指令）*

指令	作用
-release a	指定释放程序应在创建第一个存档副本后，释放存档组中的文件。如果您需要为每一个文件创建多份存档副本，请勿使用此选项。在此情况下，系统会登台第一份副本以创建第二份副本。
-release n	指定存档组中的文件永不释放。
-release p	指定释放程序应在存档文件后，部分释放存档组中的文件。

有关这些及其它 archiver.cmd 指令的详细信息，请参阅第 49 页的 “存档”。

配置释放程序

您有必要为您的站点确定高速缓存中文件的特征。如果只需登台几千字节的小文件，载入磁带是不经济的，因此您可能更希望系统将小文件保存在高速缓存中。代码示例 5-7 显示了 `releaser.cmd` 文件中用于优先释放最大文件的指令。

代码示例 5-7 优先释放最大文件的指令

```
weight_size = 1.0
weight_age = 0.0
```

此外，您可能希望将最近修改过的文件保留在高速缓存中，因为您不久可能会重新修改它们。这可以避免因登台文件以进行修改而产生的开销。在这种情况下，可使用第二组时限权数。代码示例 5-8 显示了 `releaser.cmd` 文件中用于对文件进行加权的指令，以保证严格遵守从最早修改的文件到最新修改的文件的释放顺序。

代码示例 5-8 优先释放最早修改的文件的指令

```
weight_size = .0
weight_age_access = .0
weight_age_modify = 1.0
weight_age_residence = 0.0
```

不过，大多数情况并不采用这种简单直接的方式，如下面的示例所述。

示例 1。假定您希望首先释放最大的文件。文件系统中包括成百上千个大小相同的小文件和数个大文件。小文件的大小之和可能超过一个最大文件的大小。最终，释放程序将释放所有大文件。如果指定 `weight_age = 0.0`，释放程序基本上按任意顺序释放小文件，因为它们的大小相同，并且具有相同的释放优先级。

此时，您可以设置 `weight_age = 0.01` 来进行进一步的加权。这样，如果两个文件大小相等，释放程序会首先释放较旧的文件。

示例 2。本示例介绍了一种指定如何首先释放最大文件的更好方法。

设置 `weight_size = 1.0` 和 `weight_age = 0.01`。

这两个指令通过优先选择较小的、不经常访问的文件 （而不是较大的、经常访问的文件）作为释放备选文件，因而它们违背了首先释放最大文件的原则。通过使 `weight_age` 的值更小于 `weight_size` 的值，您可以将这一影响降低到您所需的程度。例如，根据上面的设置，已登台 100 分钟的 4 KB 文件与刚刚登台的 8 KB 文件具有相同的释放优先级。

此时，释放程序会选择释放其中任何一个文件。如果它选择 4 KB 的文件，则违背了首先释放最大文件的意向。为了降低这一影响，请将 weight_age 设置为更小的值，例如 0.001。如果 4 KB 文件登台的时间为 1,000 分钟，则它与刚刚登台的 8 KB 文件具有相同的优先级。

您可以使用 no_release 和 display_all_candidates 指令以及手动运行释放程序以获得按优先级顺序排列的备选文件列表，以便根据该表来调整优先级权数。

手动运行释放程序

有时，您可能需要手动运行释放程序。要进行此项操作，您需要知道文件系统的安装点以及释放程序试图达到的下限。

例如，要释放 /sam1 文件系统中的文件，直至其占用率达到 47%，可以 root 用户身份登录，并键入以下命令：

```
# /opt/SUNWsamfs/sbin/sam-releaser /sam1 47 1.0
```

最后面的变量 weight-size 将由 releaser.cmd 文件中的 weight-size 命令改写。当释放程序运行时，它将在屏幕上显示有关信息并写入至释放程序日志文件（如果已在 releaser.cmd 文件中指定）。有关详细信息，请参阅 sam-releaser(1M) 手册页。

排除释放程序的故障

释放程序未能释放文件的原因有许多种。下面列出了一些可能的原因：

- 只有在存档文件后才能释放文件。文件可能没有存档副本。有关这一点的详细信息，请参阅第 122 页的 “文件未被存档的原因”。
- 存档程序请求不要释放文件。这可能会发生在下列情况下：
  - 存档程序刚刚登台某个离线文件以创建另一副本。
  - 已在 archiver.cmd 文件中设置 -norelease 指令，并且所有标记为 -norelease 的副本尚未存档。请注意，释放程序汇总输出显示了设定 archnodrop 标记的文件的总数。
- 文件已设置为部分释放，但文件的大小等于或小于向上圆整至磁盘分配单元 (DAU) 大小 （块大小）的部分释放大小。

- 文件在最后的 *min_residence_age* 分钟内更改了驻留状态。
- 已使用 `release -n` 命令以阻止释放目录和文件。
- `archiver.cmd` 文件中为太多的目录和文件设置了 `-release n` 选项。
- 释放程序上限设置得太高，从而造成自动释放操作发生得太迟。在 `samu`(1M) 实用程序的 `m` 显示屏幕中或使用 SAM-QFS Manager 验证这一情况，并根据需要减小该值。
- 释放程序下限设置得太高，从而造成自动释放操作停止得太早。在 `samu`(1M) 实用程序的 `m` 显示屏幕中或使用 SAM-QFS Manager 验证这一情况，并根据需要减小该值。
- 大型文件使用频繁。它们永远不会达到存档时限，永远不会存档，因此也永远不会释放。

第六章

登台

登台 是将文件数据从近线或离线存储设备复制回在线存储设备的过程。登台功能可以使您立即登台文件、永不登台文件、指定部分登台以及指定其它登台操作。例如，永不登台功能可供随机从大文件访问一小段记录的应用程序使用；启用该功能时，系统不需将文件登台为在线文件便可直接从存档介质中访问数据。

本章介绍 Sun StorEdge SAM-FS 文件登台功能。它包括下列主题：

stager.cmd 文件

可以使用 `stager.cmd` 文件指定登台程序的操作。该文件的完整路径名为 `/etc/opt/SUNWsamfs/stager.cmd`。默认情况下，登台程序将执行下列操作：

- 登台程序尝试使用库中的所有驱动器来登台文件。
- 登台缓冲区的大小由介质类型决定，并且不锁定登台缓冲区。
- 不写日志文件。
- 一次最多可以激活 1000 个登台请求。

可以使用 `stager.cmd` 文件指定有关指令来改写这些默认操作。本节的后半部分将介绍登台程序的指令。有关登台程序指令的其它信息，请参阅 `stager.cmd`(4) 手册页。

第 150 页的 “stager.cmd 文件示例”显示了编写完毕的已设置所有可能指令的 stager.cmd 文件。

代码示例 6-1 显示了本章示例中所使用的 mcf 文件。

代码示例 6-1 本章示例中使用的 mcf 文件

```
#
# Sun StorEdge SAM-FS file system configuration example
#
# Equipment       Eq Eq Family Dev Additional
# Identifier      Or Tp Set    St  Parameters
# --------------- -- -- ------ --- ----------
samfs1            60 ms samfs1
/dev/dsk/c1t1d0s6 61 md samfs1 on
/dev/dsk/c2t1d0s6 62 md samfs1 on
/dev/dsk/c3t1d0s6 63 md samfs1 on
/dev/dsk/c4t1d0s6 64 md samfs1 on
/dev/dsk/c5t1d0s6 65 md samfs1 on
#
samfs2             2 ms samfs2
/dev/dsk/c1t1d0s0 15 md samfs2 on
/dev/dsk/c1t0d0s1 16 md samfs2 on
#
/dev/samst/c0t2d0 20 od -      on
/dev/samst/c1t2u0 30 rb dog  on /var/opt/SUNWsamfs/catalog/dogcat
/dev/samst/c1t5u0 31 od dog    on
/dev/samst/c1t6u0 32 od dog    on
/dev/rmt/0cbn     40 od -      on
/dev/samst/c1t3u1 50 rb bird on /var/opt/SUNWsamfs/catalog/birdcat
/dev/rmt/2cbn     51 tp bird   on
```

▼ 创建或修改 stager.cmd 文件并应用更改

1. **使用 vi(1) 或其他编辑器来编辑 stager.cmd 文件。**

   此文件的完整路径如下所示：

```
/etc/opt/SUNWsamfs/stager.cmd
```

   有关可在这个文件中添加哪些指令的信息，请参阅以下几节：

   ■ 第 145 页的 “指定驱动器数量”

- 第 146 页的 “设置登台缓冲区大小”
- 第 147 页的 “指定日志文件”
- 第 149 页的 “指定登台请求的数量”

2. 保存并关闭 `stager.cmd` 文件。

3. 使用带 `config` 选项的 `samd`(1M) 命令应用文件更改并重新启动系统。

```
# samd config
```

指定驱动器数量

默认情况下，登台程序在登台文件时使用所有可用的驱动器。如果登台程序使所有驱动器处于繁忙状态，这会影响存档程序的活动。 `drives` 指令用于指定登台程序可用的驱动器数量。此指令的格式如下：

```
drives = library  count
```

表 6-1 *drives* 指令的变量

变量	含义
library	Sun StorEdge SAM-FS `mcf` 文件中定义的自动化库的系列集名。
count	所要使用的驱动器的最大数量。默认情况下，此数量与在 `mcf` 文件中为该库配置的驱动器数量相同。

例如，下面的指令行表示只使用 `dog` 系列集库中的一个驱动器来登台文件：

```
drives = dog 1
```

有关 `mcf` 文件的详细信息，请参阅 `mcf`(4) 手册页。

设置登台缓冲区大小

默认情况下，登台程序在将要登台的文件从存档介质恢复至在线磁盘高速缓存之前，首先将此文件读入缓冲区中的内存。可以使用 bufsize 指令来设置非默认的缓冲区大小和锁定缓冲区 （可选）。这些操作可以改善系统性能。可以尝试不同的 *buffer_size* 值以确定最适合的缓冲区大小。此指令的格式如下：

```
bufsize = media  buffer_size  [ lock ]
```

表 6-2　*bufsize* 指令的变量

变量	含义
media	指定 *mcf*(4) 手册页中列出的某存档介质类型。
buffer_size	指定一个 2 至 32 的值，默认值为 4。这个值乘以该介质类型的 *dev*_blksize 值，计算结果即为所使用的缓冲区大小。可以在 defaults.conf 文件中指定 *dev*_blksize 的值。为 *buffer_size* 指定的数值越大，所使用的内存就越多。有关此文件的详细信息，请参阅 defaults.conf(4) 手册页。
lock	lock 变量指明存档程序在登台存档副本时是否使用锁定的缓冲区。如果指定 lock，存档程序将在复制操作期间在内存中的存档缓冲区上设置文件锁定。这可以避免由于为每一个 I/O 请求锁定和取消锁定缓冲区而造成的开销，从而减少占用系统 CPU 的时间。 仅在配有大量内存的大型系统上，才有必要指定 lock 变量。如果内存不足，则可能会导致内存用尽。 只有已为需要登台的文件启用直接 I/O 时， lock 变量才有效。默认情况下，不会指定 lock 变量，并且文件系统会在所有直接 I/O 缓冲区上设置锁定 （包括用于登台的缓冲区）。有关启用直接 I/O 的详细信息，请参阅 setfa(1) 手册页、 sam_setfa(3) 库例程手册页，或 mount_samfs(1M) 手册页中介绍的 -O forcedirectio 选项。

例如，可以按以下方式在 stager.cmd 文件的指令行中指定该指令：

```
bufsize=od 8 lock
```

指定日志文件

可以要求 Sun StorEdge SAM-FS 文件系统收集文件登台事件的有关信息，并将这些信息写入日志文件。 `logfile` 指令用于指定登台程序可在其中写入记录信息的日志文件。此指令的格式如下：

```
logfile=filename [ event ]
```

其中的 *filename* 用于指定完整的路径名，

event 用于指定一个或多个登台事件。如果指定了多个 *event*，应该使用空格分隔每个 *event*。默认情况下启用以下事件：`finish cancel error`。下面列出了一些可能的 *event*：

表 6-3 *event* 变量的关键字

event	操作
`all`	记录所有登台事件。
`start`	记录文件开始登台的时间。
`finish`	记录文件结束登台的时间。默认启用。
`cancel`	记录操作员取消登台操作的时间。默认启用。
`error`	记录登台错误。默认启用。

指定了日志文件后，登台程序会将每一个登台文件的有关信息写入至日志文件中的一行或多行。该行中包括文件名、登台日期和时间以及 VSN 等信息。例如，下面的指令行指定了文件 `/var/adm/stage.log`：

```
logfile=/var/adm/stage.log
```

代码示例 6-2 显示了登台程序日志文件的示例。

代码示例 6-2 登台程序日志文件示例

```
S 2003/12/16 14:06:27 dk disk01 e.76d 2557.1759 1743132 /sam1/testdir0/filebu
1 root other root 0
F 2003/12/16 14:06:27 dk disk01 e.76d 2557.1759 1743132 /sam1/testdir0/filebu
1 root other root 0
S 2003/12/16 14:06:27 dk disk02 4.a68 1218.1387 519464 /sam1/testdir1/fileaq 1
root other root 0
S 2003/12/16 14:06:43 dk disk01 13.ba5 3179.41 750880 /sam1/testdir0/filecl 1
root other root 0
F 2003/12/16 14:06:43 dk disk01 13.ba5 3179.41 750880 /sam1/testdir0/filecl 1
root other root 0
S 2003/12/16 14:06:59 dk disk01 17.167b 1155.1677 1354160 /sam1/testdir0/filedb
1 root other root 0
F 2003/12/16 14:06:59 dk disk01 17.167b 1155.1677 1354160 /sam1/testdir0/filedb
1 root other root 0
S 2003/12/16 14:06:59 dk disk02 f.f82 3501.115 1458848 /sam1/testdir1/filecb 1
root other root 0
S 2003/12/16 14:07:15 dk disk01 1f.473 1368.1419 636473 /sam1/testdir0/fileed
1 root other root 0
S 2003/12/16 14:07:15 dk disk02 16.f15 3362.45 1065457 /sam1/testdir1/filecz 1
root other root 0
S 2003/12/16 14:07:31 dk disk01 23.201d 3005.1381 556807 /sam1/testdir0/fileeq
1 root other root 0
S 2003/12/16 14:07:47 dk disk01 26.c4d 2831.1113 1428718 /sam1/testdir0/fileez
1 root other root 0
S 2003/12/16 14:07:47 dk disk02 1b.835 3736.59 1787855 /sam1/testdir1/filedp 1
root other root 0
```

如表 6-4 所示，登台程序日志文件包含的信息行可拆分为九个字段。表 6-4 对登台程序日志文件字段的内容进行了介绍。

表 6-4 登台程序日志文件字段

字段	内容说明
1	登台活动。`S` 表示开始登台。`C` 表示取消登台。`E` 表示登台出错。`F` 表示完成登台。
2	登台操作发生的日期，格式为 *yyyy*/*mm*/*dd*。
3	登台操作发生的时间，格式为 *hh*:*mm*:*ss*。
4	存档介质类型。有关介质类型的详细信息，请参阅 `mcf`(4) 手册页。
5	VSN。

表 6-4　登台程序日志文件字段　*（接上页）*

字段	内容说明
6	存档文件 （tar(1) 文件）在介质上的起始物理位置和存档文件中的文件偏移量（采用十六进制表示）。
7	索引节点编号和世代编号。世代编号是在索引编号被重新使用后生成的一个附加编号，它与索引编号一起用来标识使用的唯一性。
8	文件的大小。
9	文件的名称。
10	存档副本数。
11	文件的用户 ID。
12	文件的组 ID。
13	请求者的组 ID。
14	待登台文件所在驱动器的设备序号。

指定登台请求的数量

可以使用 maxactive 指令来指定一次可以执行的登台请求的数量。此指令的格式如下：

```
maxactive=number
```

默认情况下， *number* 的值为 4000。所允许的最小值为 1。

例如，下面的指令行指定队列中最多可以同时存在 500 个登台请求。

```
maxactive=500
```

stager.cmd 文件示例

代码示例 6-3 显示了 stager.cmd 文件的一个示例。

代码示例 6-3 stager.cmd 文件示例

```
# This is stager.cmd file /etc/opt/SUNWsamfs/stager.cmd
drives=dog 1
bufsize=od 8 lock
logfile=/var/adm/stage.log
maxactive=500
```

archiver.cmd 文件在登台过程中的作用

archiver.cmd 文件中的大多数指令会影响存档过程，但存档组分配指令可以使您指定应用于存档组中所有文件的登台属性。存档组分配指令的格式如下：

archive_set_name path [*search_criteria ...*] *directives ...*]

第 49 页的 “存档”一章中详细地介绍了存档组分配指令及其变量。表 6-5 显示了存档组分配指令中可用作登台指令的 *directives （指令）*。

表 6-5 archiver.cmd 文件中的登台 *directives （指令）*

指令	作用
-stage a	指定存档组中的文件应联合登台。
-stage n	指定存档组中的文件永不登台。

有关这些指令及其它 archiver.cmd 指令的详细信息，请参阅第 49 页的 “存档”。

使用 preview.cmd 文件确定预备请求的优先级

存档程序和登台程序进程均可请求载入及卸载介质。如果请求的数量超过可用于介质载入的驱动器数量，则多余的请求会发送至预备队列。

预备队列中的存档和登台请求是指那些无法立即满足的请求。默认情况下，系统按先进先出 (FIFO) 的顺序执行预备请求。

预备队列中的条目数量取决于 defaults.conf 文件中的 previews= 指令。有关更改该指令值的信息，请参阅 defaults.conf(4) 手册页。

可以为各个预备请求分配不同的优先级。可以通过在预备命令文件中输入指令来改写 FIFO 默认值，此预备命令文件的位置如下：

```
/etc/opt/SUNWsamfs/preview.cmd
```

此文件根据请求是用于登台还是存档来安排预备请求。也可提高特定 VSN 的优先级。此外， preview.cmd 文件中的设置也可根据上限 (HWM) 或下限 (LWM) 设置，来重新分配所有或特定文件系统的预备请求的优先级。

sam-amld 后台程序在启动时读取预备请求指令。这些指令必须是每一条独占一行。如果在 sam-amld 后台程序正在运行时更改此文件，那么必须重新启动 sam-amld 后台程序，以使更改生效。注释行以井字符 (#) 开头，并且延伸至行的末尾。有关此文件的详细信息，请参阅 preview.cmd(4) 手册页。

preview.cmd 文件可以包含以下两种类型的指令：

- 全局指令，应用于所有文件系统。这些指令必须出现在第一行 fs = 前。
- 专用于文件系统的指令，它位于全局指令的后面。与 archiver.cmd 文件相似， preview.cmd 文件可以包含专用于单个文件系统的指令。在此文件中，专用于单个文件系统的指令必须出现在所有全局指令的后面。

  这些文件系统指令必须以 fs = *file_system_name* 指令开头。该指令用于指定文件系统，其后的所有指令均应用于该文件系统。文件中可以包含多个文件指令块。文件系统指令的应用范围到出现下一行 fs = 或到达文件末尾为止。

注意 – 当多条指令影响一个文件系统时，专用于文件系统的指令将取代全局指令。

VSN 和时限指令 （全局）

VSN 和时限优先级指令均是全局指令。如果在 preview.cmd 文件中指定这些指令，则它们必须出现在任何专用于文件系统的指令的前面。也就是说，它们必须出现在所有 fs = 指令的前面。 VSN 优先级指令的格式如下：

```
vsn_priority = value
```

该指令是一个静态的优先级因子，它表示具有高优先级的 VSN 的总优先级将要增加的值。 vsn_priority 的默认值为 1000.0。当 VSN 被安排为预备请求时，它们必须已被设置优先级标记，才能获得此增加值。可以使用带 p 选项的 chmed(1M) 命令，来设置优先级标记 （例如 chmed +p lt.AAA123）。只有提交的 VSN 请求尚不是预备请求时，对它们设置此优先级标记才会有效。时限优先级指令的格式如下：

```
age_priority = factor
```

该指令是一个静态的优先级因子，但它的整体影响是动态的。 age_priority 因子与请求成为预备请求的秒数 （即请求等待的时间）相乘，其结果将与请求的总优先级相加。请求的等待时间越长，时限因子就越大。设置该因子有助于确保较旧的请求，不会被具有其它较高优先级因子的较新请求无限期地取代。

如果该因子大于 1.0，则它可以增加时间因子在计算总优先级中的重要性。如果小于 1.0，则会降低时间因子的重要性。如果将该因子设置为 0.0，则在计算总优先级级时不考虑时间因子。

对于没有设置优先级标记的 VSN，系统将根据它在队列中等待的时间提高其优先级。这样，该 VSN 的优先级可能会高于以后进入队列且已设置优先级标记的 VSN。

水印指令 （全局或文件系统专用）

预备请求的水印指令既可以用作全局指令，也可用作文件系统的专用指令。水印优先级指令用于确定预备请求的水印优先级 (wm_priority)。代码示例 6-4 表明 wm_priority 因子是多个设定值的和。

代码示例 6-4 wm_priority 的计算方法

```
lwm_priority +
lhwm_priority +
hlwm_priority +
```

代码示例 6-4 wm_priority 的计算方法 *（接上页）*

```
hwm_priority
_____________________
= wm_priority
```

当 wm_priority 因子是正数时，计算出的总优先级所产生的影响是：提高存档请求的优先级而降低登台请求的优先级。不过， wm_priority 因子还可以是负数。在此情况下，将会降低存档请求的总优先级，这导致系统优先处理登台请求，然后再处理存档请求。如果将此因子设置为 0.0 （或根本不指定命令），则表示在文件系统遇到此情况时，系统不会对存档请求采取任何特殊的操作。有关此因子的详细信息，请参阅第 155 页的 “示例 1：强制执行登台请求”中的示例。

表 6-6 显示了四个水印优先级指令及其变量。

表 6-6 水印优先级指令

优先级指令	变量
lwm_priority = *value*	其中的 *value*，用于指定当文件系统低于下限 (LWM) 时，存档请求的 wm_priority 因子的改变量。默认设置为 0.0。
lhwm_priority = *value*	其中的 *value*，用于指定当文件系统从超过下限 (LWM) 上升至下限以上，但仍然低于上限 (HWM) 时，存档请求的 wm_priority 因子的改变量。这通常表示文件系统中的文件正在增加。默认设置为 0.0。
hlwm_priority = *value*	其中的 *value*，用于指定当文件系统从超过上限 (HWM) 降至上限以下，但仍然低于下限 (LWM) 时，存档请求的 wm_priority 因子的改变量。这通常表示释放程序不能释放足够的磁盘空间，以使文件系统低于下限 (LWM)。默认设置为 0.0。
hwm_priority = *value*	其中的 *value*，用于指定当文件系统超过上限 (HWM) 时，存档请求的 wm_priority 因子的改变量。默认设置为 0.0。

总之，这四个水印设置用于创建包括百分比值 （表示文件系统的占用率）以及 HWM 和 LWM 设置级别在内的动态优先级因子。分配给预备请求的值取决于优先级因子是全局性的，文件系统专用的，还是未设置。

当文件系统的情况发生变化时，系统将根据以下条件重新计算与该文件系统关联的每一个 VSN 的优先级：相应的水印优先级设置；是否使用 chmed(1M) 命令的 p 选项设置了优先级。

水印优先级仅用于计算与存档有关的介质请求，而不能用于计算与登台有关的介质请求。

以下示例指令显示了如何在文件系统处于 HLWM 位置时，小幅度地提高存档请求的优先级。代码示例 6-5 显示了一些设置，这些设置可用于启用释放程序以释放足够的磁盘空间，从而使文件系统低于 LWM。

代码示例 6-5 使文件系统低于 LWM 的设置

```
lhwm_priority = -200.0
hlwm_priority = 100.0
```

计算预备请求的总优先级

预备请求的优先级数值由数个静态和动态因子共同决定。数值越大，优先级就越高。静态优先级因子在生成请求时进行设置。一旦请求生成并进入等待执行状态，静态优先级因子对总优先级的影响将不会发生变化。在请求等待执行期间，动态优先级因子可以提高或降低请求的总优先级。

预备请求的总优先级是所有优先级因子的总和，其计算方式如下：

```
total priority = vsn_priority + wm_priority + (age_priority *
time_in_sec_as_preview_request)
```

如何设置预备请求的优先级方案

除非绝对需要，否则请勿更改默认的预备请求 FIFO 方案。在下列条件下，可能需要更改默认的预备请求 FIFO 方案：

- 条件 1：确保首先处理登台请求，然后处理存档请求。
- 条件 2：确保存档请求在文件系统将要充满时获得最高优先级。
- 条件 3：将使用特定介质组的请求排在预备请求列表的顶部。

对于用户对数据访问要求极高、 VSN 驱动器数受限制或将文件存档作为后台功能的环境，可以使用 `preview.cmd` 文件来改变存储系统资源执行登台请求的方式。可以对 `preview.cmd` 文件的设置进行自定义，以支持上述所有方案，并影响已配置的 Sun StorEdge SAM-FS 环境。

由于该文件中的设置对数据并无影响，因此建议尝试并调整不同的指令设置，以便在权衡每一个预备请求的优先级时，能够在存档请求和登台请求之间找到适当的平衡点。

代码示例 6-6 显示了可以满足上述三个条件的 preview.cmd 文件。

代码示例 6-6 preview.cmd 文件示例

```
# condition 1
lwm_priority = -200.0
lhwm_priority = -200.0
hlwm_priority = -200.0
# condition 2
hwm_priority = 500.0
# condition 3
age_priority = 1.0
```

示例 1：强制执行登台请求

下面的设置示例提供了一种可以确保登台请求先于存档请求处理的方法。本示例假定：

- 队列中已经有多个请求的等待时间达到了 100 秒。
- vsn_priority 的默认值为 1000。

表 6-7 显示了如何计算请求的总优先级。

表 6-7 请求优先级计算示例

优先级	计算
具有优先级的存档 VSN， LWM：	1000 + (-200) + (1 x 100) = 900
具有优先级的登台 VSN， LWM：	1000 + 0 + (1 x 100) = 1100
无优先级的登台 VSN， LWM：	0 + 0 + (1 x 100) = 100

本示例表明在其它因子都相同时， wm_priority 为负值会使系统优先处理登台请求，然后再处理存档请求。

示例 2：强制执行存档请求

当在环境中权衡将文件登台回用户与将新文件存档至介质这二者的重要性时，最为关心的是超过上限 (HWM) 的情形。在此情况下，如果没有足够的满足存档要求的文件来降低文件系统的占用率，则完成待定的存档请求是防止文件系统充满的次佳方法。

在此情况下，只需在 preview.cmd 文件中进行以下设置：

```
hwm_priority = 500.0
```

示例 3：根据介质确定请求的优先级

在面向项目的环境中，特定用户可能只处理占用特定 VSN 的文件组，并且其数据与其他用户的数据相互分开。在这种环境中，某些项目有时可能具有较高的优先级；因此，它们需要具有优先使用可用系统存储资源的权利。这时可以使用下面的指令配置 preview.cmd 文件，以适当给予用户及其介质优先使用介质驱动器的权利：

```
hwm_priority = 5000.0
```

然后，对于每一个属于优先级用户组的 VSN，输入下面的信息：

```
# chmed +p lt.AAA123  ## or whatever VSN is used
```

以后，每一个要求访问 VSN AAA123 （或使用的任何 VSN）的请求均优先于预备队列中的其它待定安装请求。

将来，若要降低用户介质的优先级，请为每一个 VSN 输入相反的命令：

```
# chmed -p lt.AAA123  ## or whatever media type is used
```

示例 4：确定复杂请求的优先级

假设存在两个具有以下要求的 Sun StorEdge SAM-FS 文件系统：

- 请求在队列中的等待时间不能太长 (age_priority)。
- 当文件系统低于下限 (LWM) 时，优先处理登台请求。
- 当文件系统高于下限 (LWM) 且低于上限 (HWM) 时，无需区分存档请求或登台请求的优先级。代码示例 6-7 显示了受影响的指令。

代码示例 6-7　　指令

```
lwm_priority =  -200.0
lhwm_priority = 0.0
hlwm_priority = 0.0
```

在此情况下，其它指令保持不变。

当文件系统超过上限 (HWM) 时，优先处理存档请求。

如果两个文件系统同时超过了上限 (HWM)，则应首先防止第二个文件系统 （例如 samfs2）充满。例如，当 samfs1 是一个用户工作文件系统，而 samfs2 是一个关键文件系统时，便会出现这种情况。

在任何情况下 （不管出现何种情形），如果已设置 chmed(1M) 命令的 p 标记，则要求访问选定 VSN 组的请求将优先于预备请求队列中的其它请求。

代码示例 6-8 显示了可根据前面列出的要求确定请求优先级的 preview.cmd 文件。

代码示例 6-8 preview.cmd 文件

```
age_priority = 100.0
vsn_priority = 20000.0
lhwm_priority = -200.0
hlwm_priority = -200.0
fs = samfs1
hwm_priority = 1000.0
fs = samfs2
hwm_priority = 5000.0
```

第七章

回收

回收 是指从存档卷中收回空间的过程。回收程序与存档程序配合工作，以收回由无用的存档副本占用的空间。当用户修改某个文件时，即可从系统中清除与该文件的旧版本相关联的存档副本。回收程序可以识别那些其中绝大部分是过期存档副本的存档卷，并将这些卷中的非过期副本移动到其它卷中。当某个给定的卷中只包含过期副本时，即可执行站点定义的操作。例如，可以重新标记此类卷以便立即重新使用此类卷，或将其中的数据导出至离站存储设备，从而单独保存文件更改的历史记录。由于回收过程只与用户的数据文件有关，因此用户不会觉察到回收过程。

本章包括下列主题：

回收程序概述

回收程序负责将过期存档副本占用的空间，保持在由站点指定参数定义的最低水平。在任何时候，给定存档卷的空间均由以下各项组成：

- *当前数据空间*，由当前有效的存档映像占用的空间。
- *过期数据空间*，由当前不再有效的存档映像占用的空间。
- *可用空间*，未被当前有效或过期的存档映像占用的空间。

卷的 *容量* 是指卷中可用于存储数据的总空间量。例如，对于一个已写入 3 GB 数据的 10 GB 磁带卷来说，它的容量为 10 GB，可用空间为 7 GB。

对于全新的存档介质或新标记的存档介质，其容量等于可用空间。当将数据存档至该介质时，可用空间会减少，而当前数据空间会增加。

当更改或删除文件系统中已存档的文件时，这些文件的存档映像会过期，并且其类别由当前数据类别变为过期数据类别。这些映像占用的物理空间并没有发生变化；只是文件系统中已没有指向该空间的文件。

这些过期的映像 （即过期数据）最终会占满全部可用空间。只有回收空间，才能删除这些映像并使它们占用的空间变为可用空间。回收程序的目标是将过期数据占用的空间转变为可用空间，而丝毫不损坏任何当前数据。

例如，只能在可移动介质卡盒 （如磁带）中添加数据，而不能在其中重新写入数据。重新使用卡盒的唯一方法是，从卡盒中移走所有当前数据，重新标记卡盒，然后从头开始使用卡盒。

通过输入 `sam-recycler`(1M) 命令来启动回收过程。回收过程既可手动执行，也可通过 `cron`(1) 作业来执行。表 7-1 介绍了回收方法。

表 7-1 回收方法与介质类型

回收方法	介质和说明
按自动化库	可移动介质卡盒。 当按库进行存档时，可以在 `recycler.cmd` 文件中指定回收指令。
按存档组	可移动介质卡盒和磁盘。 当按存档组进行存档时，不要使用 `recycler.cmd` 文件。而应将所有回收指令放置在 `archiver.cmd` 文件中。

注意，既可以按库进行回收，也可以按存档组进行回收。如表 7-1 所示，如果存档至磁盘，那么只能按存档组进行回收。

回收程序和存档程序协同工作，具体如下：

1. 回收程序使用 `rearchive` 属性来标记卷中所有当前 （有效）的存档映像。
2. 如果是将文件存档至可移动介质，那么回收程序使用 `recycle` 属性来标记所选择的存档卷。这可阻止存档程序再向此卷中写入存档映像。
3. 存档程序将所有已做标记的映像移至另一个卷。此操作过程称为*重新存档*。当存档程序将当前的存档映像从旧卷移至新卷后，旧卷中只包含可用空间和过期数据空间。如果是存档至可移动介质卡盒，那么可以重新标记并重新使用该卡盒。如果是存档至磁盘，那么回收程序将删除包含已过期存档映像的文件。

回收程序可以定期运行。每次启动该程序后，它会尽力完成所有工作。在存档程序可以重新存档文件之前，回收程序必须完成要重新存档的副本的标记工作。

有时，设置了 `rearchive` 属性的已过期的存档映像仍会保留在介质中。这可能会发生在下列情况下：

- 在回收程序标记已过期的存档映像后，存档程序没有运行。
- 在移动未过期的存档映像时，存档程序没有可使用的介质。
- 存档程序出现了其它各种异常。

在两次运行期间，回收程序将状态信息保留在库目录和索引节点中。在回收过程中，可以使用 sls(1) 命令及其　选项来显示文件的相关信息。 sls(1) 命令的输出可以显示，是否已经安排对某个文件进行重新存档。

回收指令

recycler.cmd 文件可以接受的指令如以下几节所述：

- 第 161 页的 “指定日志文件：logfile 指令”
- 第 161 页的 “阻止回收：no_recycle 指令”
- 第 162 页的 “指定回收整个自动化库：库指令”

指定日志文件：logfile 指令

logfile 指令用于指定回收程序日志文件。此指令的格式如下：

```
logfile = filename
```

其中的 *filename*，用于指定日志文件的路径。

下面是 logfile= 指令行的一个示例：

```
logfile=/var/adm/recycler.log
```

阻止回收：no_recycle 指令

no_recycle 指令可以阻止卷的回收。要指定 VSN，可以使用正则表达式以及一个或多个特定介质类型。此指令的格式如下：

```
no_recycle media_type VSN_regex [ VSN_regex ... ]
```

表 7-2　no_recycle 指令的变量

变量	含义
media_type	指定 mcf(4) 手册页中列出的某一介质类型。
VSN_regexp	指定一个或多个由以空格分隔的正则表达式所描述的卷。有关正则表达式的格式信息，请参阅 regexp(5) 手册页或第 76 页的 “文件大小 search_criteria：-minsize 和 -maxsize”。

指定 *media_type*，可以阻止回收存储在特定介质类型上的卷。可以通过指定一个或多个 *VSN_regexp* 参数值，使用正则表达式来标识那些免于回收的特定卡盒。

例如，下面的指令行可阻止回收程序回收那些 VSN 标识以 DLT 开头的磁带卷：

```
no_recycle lt DLT.*
```

指定回收整个自动化库：库指令

库指令可用于为那些与特定库关联的 VSN 指定各种不同的回收参数。此指令的格式如下：

```
library  parameter  [  parameter  ...  ]
```

其中的 *library*，用于指定库名称，该名称与在 mcf(4) 文件的系列集字段中指定的名称相同。

而其中的 *parameter*，用于指定一个或多个以空格分隔的 *parameter* 关键字 （见表 7-3）。

表 7-3　库指令中 *parameter* 的值

parameter	操作
-dataquantity *size*	限制回收程序可以安排重新存档的数据量，以免清除有用数据所在的卷。默认值为 1 GB。
-hwm *percent*	库利用率的上限。默认值为 95。
-ignore	阻止回收库中的卷。在测试 recycler.cmd 文件时，此指令十分有用。

表 7-3　库指令中 *parameter* 的值　*（接上页）*

parameter	操作
`-mail [ email_address ]`	向指定的 *email_address* 发送电子邮件消息。默认情况下，系统不发送电子邮件。如果已指定 `-mail` 但未带任何变量，则会将电子邮件发送至 `root` 用户。
`-mingain value`	最小 VSN 增益百分比。默认值为 50。
`-vsncount count`	限制可回收卷的数量。默认值为 1。

以下面的指令行为例：

```
gr47 -hwm 85 -ignore -mail root -mingain 40
```

它为库 `gr47` 指定了以下各项：

- 当库中卷的占用率达到 85% 时，应考虑对库执行回收操作。
- 最小增益百分比为 40%。
- 重新存档的数据量最多为 1 GB。这是一个默认值，因此未在 `recycler.cmd` 文件中指定。
- 只能回收一个卷。这也是一个默认设置。
- 将回收消息发送至 `root` 用户。

配置回收程序

配置回收程序之前，请注意以下事项：

- `archiver.cmd` 文件中的指令按存档组控制回收过程。 `recycler.cmd` 文件中的指令按库控制回收过程。此外， `recycler.cmd` 文件还控制着回收程序的一般操作。有关回收程序指令的信息，请参阅第 161 页的 “回收指令”。
- 不要回收包含可移动介质文件的卷。可移动介质文件可以使用 `request`(1) 命令来创建。回收程序不会保留由 `request`(1) 命令创建的可移动介质文件。包含可移动介质文件的卷永远不能清除干净。
- 当 Sun StorEdge SAM-FS 文件系统正在执行维护操作时，不要运行回收程序。回收程序使用 `.inodes` 文件和 `mcf` 文件，来识别当前文件或过期文件以及与文件系统关联的设备。如果没有这些文件中的正确信息，回收程序可能会将当前存档数据视为过期数据而对其执行回收操作。

- 所有 Sun StorEdge SAM-FS 文件系统必须都已安装，方可运行回收程序。如果要对在线磁盘执行回收操作，则必须安装包含此磁盘卷的文件系统，并且主机系统必须可访问。

默认情况下，系统不会启用回收程序。必须通过输入 sam-recycler(1M) 命令来启动回收过程。启动回收程序后，第 162 页的 “指定回收整个自动化库：库指令” 中指定的默认回收程序设置将会生效。有关回收程序的详细信息，请参阅 sam-recycler(1M) 手册页。

以下几节介绍回收程序的配置过程。此过程包括下列步骤：

- 第 164 页的 “步骤 1：创建 recycler.cmd 文件 （可选）”
- 第 167 页的 “步骤 2：编辑 archiver.cmd 文件 （可选）”
- 第 168 页的 “步骤 3：运行回收程序”
- 第 169 页的 “步骤 4：为回收程序创建 crontab 文件 （可选）”
- 第 170 页的 “步骤 5：删除 -recycle_ignore 和 ignore 参数”
- 第 170 页的 “步骤 6：创建 recycler.sh 文件 （可选）”

如果要回收库中的卡盒，则配置过程中还需创建 recycler.cmd 文件以及 （可选）编辑 archiver.cmd 文件。如果将文件存档至磁盘，则只能按存档组进行存档，因此要启用这些磁盘卷的回收过程，还需要编辑 archiver.cmd 文件。下面的过程说明了如何为存档介质配置回收程序。

▼ 步骤 1：创建 recycler.cmd 文件 （可选）

如果要回收库中卡盒上的存档副本，请执行本步骤。

如果要回收磁盘卷上的存档副本，则不必执行本步骤，因为回收过程由 archiver.cmd 文件中的指令控制。有关在 archiver.cmd 文件中配置回收过程的详细信息，请参阅第 167 页的 “步骤 2：编辑 archiver.cmd 文件 （可选）”。

recycler.cmd 文件中包含了通用的回收指令，也可包含针对 Sun StorEdge SAM-FS 环境中每一个库的指令。有关回收指令的信息，请参阅第 161 页的 “回收指令”。

即使按存档组执行回收操作，也应在 recycler.cmd 文件中配置每一个库。这可以确保回收程序能够回收不属于存档组的 VSN （如有必要）。

典型的 recycler.cmd 文件中包含以下指令行：

- logfile= 指令行，用于指定回收程序日志文件。系统将回收消息和回收报告写入至该文件。
- 针对每一个含有待回收卷的库的一行或多行指令。该指令行必须包含要回收的库的系列集名 （来自 mcf 文件）。这使回收程序可以识别库。

由于仍在创建 recycler.cmd 中的指令行，并且尚未经过测试，因此应使用 ignore 关键字。本过程后面的步骤中会指导您删除 ignore 关键字。

要创建 recycler.cmd 文件，请执行以下步骤：

1. 成为超级用户。
2. 使用 vi(1) 或其它编辑器打开文件 /etc/opt/SUNWsamfs/recycler.cmd。
3. 添加本章所述的一行或多行指令来控制回收程序的活动。
4. 保存并关闭该文件。

recycler.cmd 文件示例

代码示例 7-1 显示了 recycler.cmd 文件的示例。

代码示例 7-1 recycler.cmd 文件示例

```
logfile = /usr/tmp/recycler.log
stk30  -hwm 51  -mingain 60  -ignore  -mail  root
```

以下几小节介绍代码示例 7-1 中指定的参数。

-hwm 51 *参数*

通过指定使用率的上限，可使得当介质使用率低于此上限时，回收过程不能启动。此百分比是指库中已用空间与总容量的比率。例如，某个库含有 10 个 20 GB 的磁带，其中三个磁带的使用率为 100%，另外七个磁带的占用率均为 30%，则介质利用率为：

```
((3* 1.00 + 7 * 0.30) * 20G ) / ( 10 * 20G ) * 100%= 51%
```

请注意，该计算方法并不区分当前数据和过期数据，它只考虑介质的使用量。

在本示例中，如果上限为 51% 或更小，回收程序不会自动选择自动化库的任何 VSN 进行回收。

注意 – 可以使用下列命令设置回收标记，从而强制回收某 VSN：

chmed +c lt.AAA123

设置 +c 回收标记后，存档程序不会再向该卷中写入任何存档映像。可以通过 samu(1M) 实用程序来查看 +c 标记。有关详细信息，请参阅 chmed(1M) 和 samu(1M) 手册页。

-mingain 60 *参数*

minimum VSN gain percentage 用于设置通过回收卡盒所获得的空间增加量的下限。例如，在自动化库的某个卡盒中， 95% 为当前数据空间，另外 5% 是过期数据空间，因此通过回收该卡盒所获得的增益百分比仅为 5%。为获得此空间 （即 5%）而移动另外 95% 的空间，这可能是不值得的。将请最小增益百分比设置为 6% 或更大值，可以防止回收程序自动选择此示例 VSN 进行回收。

另一个示例是某个卡盒中具有 90% 的过期数据空间， 5% 的当前数据空间，以及 5% 的可用空间。如果回收此卡盒，则会获得 90% 的增益。

-ignore *参数*

-ignore 参数用于防止回收程序回收某个特定的库，在配置回收程序时，应使用此关键字。

-mail root *参数*

-mail 关键字用于指定回收程序在回收指定的库后发送电子邮件。邮件消息的主题行如下所示：

```
Robot robot-name recycle
```

表 7-2 显示了消息正文示例。

代码示例 7-2 回收消息示例

```
我将回收 VSN vsn。
未在此 media changer 中找到任何备选 VSN。
正在处理先前选定的 VSN vsn。
Previously selected VSN vsn is now finished recycling. It will now
be post-recycled.
```

▼ 步骤 2：编辑 archiver.cmd 文件（可选）

如果按存档组执行回收，请执行本步骤。如果将文件存档至磁盘，则按存档组执行回收是唯一可行的方法，因此必须完成本步骤才能进行回收。

如果按库执行回收，请继续下一步骤。

- **要编辑 archiver.cmd 文件，请执行第 61 页的“确定是否需要编辑 archiver.cmd 文件，或是否需要编辑临时的 archiver.cmd 文件。（可选）”所述步骤。**

  archiver.cmd 文件中添加的用于启用按存档组回收的指令，必须位于 params 和 endparams 指令之间。表 7-4 显示了可使用的存档组回收指令。

表 7-4 存档组回收指令

指令	功能
-recycle_dataquantity *size*	限制回收程序可以安排重新存档的数据量，以免清除有用数据所在的卷。
-recycle_hwm *percent*	设置上限百分比。
-recycle_ignore	阻止回收存档组。
-recycle_mailaddr *mail_address*	将回收程序消息发送到 *mail_address*。
-recycle_mingain *percent*	将回收对象限制为那些按 *percent* 或更大百分比增加可用空间的 VSN。
-recycle_vsncount *count*	将可重新存档的卷的数量限制为 *count*。

有关上述指令的详细信息，请参阅第 49 页的 “存档”或 archiver.cmd(4) 手册页。

代码示例 7-3 显示了用于回收磁盘存档的 archiver.cmd 示例。

代码示例 7-3 archiver.cmd 文件中的磁盘存档指令

```
fs = samfs1
     1 2m

arset0  testdir0
     1 2m
     2 4m

arset1  testdir1
     1 2m
     2 4m
```

代码示例 7-3 archiver.cmd 文件中的磁盘存档指令 *（接上页）*

```
params
arset0.1 -disk_archive disk01 -recycle_hwm 80 \
    -recycle_mingain 20 -recycle_ignore
arset1.1 -disk_archive disk02 -recycle_hwm 80 \
    -recycle_mingain 20 -recycle_ignore
endparams
```

▼ 步骤 3：运行回收程序

1. **运行 sam-recycler(1M) 命令。**

   回收程序将读取 recycler.cmd 文件。

2. **检查标准输出、日志、 SAM 日志和 /var/adm/messages，看看是否存在来自回收程序的错误消息。**

   如果有错误则请更正。

   代码示例 7-4 显示了用于回收可移动介质卡盒的回收程序的日志文件示例。

代码示例 7-4 可移动介质卡盒的回收程序日志文件示例

```
========== Recycler begins at Wed Dec 12 14:05:21 2001 ==========
Initial 2 catalogs:

0  Family: m160                  Path: /var/opt/SUNWsamfs/catalog/m160
   Vendor: ADIC                  Product: Scalar 100
   SLOT                  ty    capacity         space vsn
      0                  at        25.0G        25.0G CLN005
      1                  at        48.5G         6.1G 000003
      2                  at        48.5G        32.1G 000004
      3                  at        48.5G        35.1G 000005
      4                  at        48.5G        44.6G 000044
      5                  at        48.5G        45.1G 000002
      6                  at        48.5G        45.9G 000033
      7                  at        48.5G        48.5G 000001
   Total Capacity:  364.8G bytes, Total Space Available: 282.3G bytes
   Volume utilization 22%, high 95% VSN_min 50%
   Recycling is ignored on this robot.




1  Family: hy                    Path: /var/opt/SUNWsamfs/catalog/historian
   Vendor: Sun SAM-FS            Product: Historian
   SLOT                  ty    capacity         space vsn
```

代码示例 7-4　可移动介质卡盒的回收程序日志文件示例 *（接上页）*

```
      (no VSNs in this media changer)
   Total Capacity:  0    bytes, Total Space Available: 0    bytes
   Volume utilization 0%, high 95% VSN_min 50%
   Recycling is ignored on this robot.



8 VSNs:

                     ---Archives---   -----Percent-----   m160
 ----Status-----     Count    Bytes   Use Obsolete Free   Library:Type:VSN
no-data VSN             0        0     0    87     13    m160:at:000003
no-data VSN             0        0     0    33     67    m160:at:000004
no-data VSN             0        0     0    27     73    m160:at:000005
no-data VSN             0        0     0     8     92    m160:at:000044
no-data VSN             0        0     0     7     93    m160:at:000002
no-data VSN             0        0     0     5     95    m160:at:000033
empty VSN               0        0     0     0    100    m160:at:CLN005
empty VSN               0        0     0     0    100    m160:at:000001



Recycler finished.

========== Recycler ends at Wed Dec 12 14:05:32 2001 ===========
```

代码示例 7-5 显示了用于回收磁盘存档文件的回收程序的日志文件示例。

代码示例 7-5　磁盘存档文件的回收程序日志文件示例

```
---Archives---    -----Percent-----
 ----Status-----     Count     Bytes    Use Obsolete Free    Library:Type:VSN
new candidate           0       0        0     41     59   <none>:dk:disk01


677 files recycled from VSN disk01 (mars:/sam4/copy1)
0 directories recycled from VSN disk01 (mars:/sam4/copy1)
```

▼ 步骤 4：为回收程序创建 crontab 文件 （可选）

如果系统运行正常，即可为超级用户创建 crontab 条目，以定期运行回收程序。您可能希望每隔 2 个小时运行一次回收程序，这视您站点的情况而定。

- **创建 crontab 条目。**

  有关的信息，请参阅 cron(1M) 手册页。

  下面是 root 用户的 crontab 文件中的条目示例，它确保 cron 后台程序在每个奇数小时内每隔 5 分钟运行一次回收程序：

```
5 1,3,5,7,9,11,13,15,17,19,21,23   * * * /opt/SUNWsamfs/sbin/sam-recycler
```

▼ 步骤 5：删除 -recycle_ignore 和 ignore 参数

1. **使用 vi(1) 或其它编辑器删除 archiver.cmd 文件中的 -recycle_ignore 参数。**
2. **使用 vi(1) 或其它编辑器删除 recycler.cmd 文件中的 ignore 参数。**

   现在，回收程序便真正开始执行回收过程。

▼ 步骤 6：创建 recycler.sh 文件 （可选）

如果要回收可移动介质卡盒上的存档副本，请执行本步骤。如果只将文件存档至磁盘，则不要执行本步骤。

在存档程序将 VSN 中的所有当前映像重新存档至另一个 VSN 后，回收程序将执行 recycler.sh 脚本。有关示例，请参阅 recycler.sh(1M) 手册页。可以在 /opt/SUNWsamfs/examples/recycler.sh 中找到另一个示例，它显示了如何重新标记已回收的 VSN 并向超级用户发送电子邮件。

回收程序使用以下变量调用 /opt/SUNWsamfs/sbin/recycler.sh 脚本：

```
Media type: $1  VSN: $2  Slot: $3  Eq: $4
```

当回收程序确定已清除 VSN 中所有已知的有效存档副本后，它将调用 /opt/SUNWsamfs/sbin/recycler.sh 脚本。您应确定您的站点对清除已回收卡盒的要求。某些站点选择重新标记或重新使用卡盒；而其它站点选择从自动化库中取出卡盒以便将来用于访问历史文件。有关详细信息，请参阅 recycler(1M) 和 recycler.sh(1M) 手册页。

排除回收程序的故障

回收程序的最常见问题是在启动时显示与下例类似的消息：

```
正在等待 VSN mo:OPT000 清除，它仍具有 123 个有效存档副本。
```

可能会导致回收程序生成此消息的情况有：

- 情况 1：存档程序未能重新存档卷中的 123 个有效存档副本。
- 情况 2：这 123 个存档副本不指向文件系统中的文件，而是指向 123 个元数据存档副本。

造成第 1 种情况的原因可能包括：

- 需要重新存档的文件被标记为 no_archive。
- 需要重新存档的文件位于 no_archive 存档组中。
- 由于没有可用的 VSN 而无法存档文件。
- archiver.cmd 文件中包含 wait 指令。

要确定问题是由哪种情况引起的，请运行回收程序并选择 -v 选项。如代码示例 7-6 所示，此选项将显示与回收程序日志文件中这 123 个存档副本相关联的文件的路径名。

代码示例 7-6　　回收程序消息

```
/sam/fast/testA 的存档副本 2 位于 VSN LSDAT1 上
/sam3/tmp/dir2/filex 的存档副本 1 位于 VSN LSDAT1 上
无法查找文件系统 /sam3 的路径名（inum/gen 30/1）的存档副本 1 位于 VSN
LSDAT1 上
/sam7/hgm/gunk/tstfilA00 的存档副本 1 位于 VSN LSDAT1 上
/sam7/hgm/gunk/tstfilF82 的存档副本 1 位于 VSN LSDAT1 上
/sam7/hgm/gunk/tstfilV03 的存档副本 1 位于 VSN LSDAT1 上
/sam7/hgm/gink/tstfilA06 的存档副本 1 位于 VSN LSDAT1 上
/sam7/hgm/gink/tstfilA33 的存档副本 1 位于 VSN LSDAT1 上
正在等待 VSN dt:LSDAT1 清除，它仍具有 8 个有效存档副本。
```

本输出示例中显示了包含七个路径名的消息和一个包含*无法查找 . . . 路径名*文本的消息。要纠正未清除 LSDAT1 的问题，需要确定未重新存档这七个文件的原因。重新存档这七个文件之后，只有一个存档副本未与文件关联。请注意，只有在系统崩溃而造成 .inodes 文件部分损坏时才会发生这种情况。

要解决查找路径名的问题，请运行 samfsck(1M) 以收回遗留索引节点。如果您不选择运行 samfsck(1M)，或您无法卸载文件系统以运行 samfsck(1M)，请检查 recycler -v 的输出，以确保清除了有效存档副本，然后手动重新标记卡盒。不过，由于回收程序会再次遇到仍保留在 .inodes 文件中的无效索引节点，因此当该 VSN 再次成为回收对象时，会发生同样的问题。

当回收程序未能选择任何 VSN 以进行回收时，会发生另一种回收程序问题。要确定每一个 VSN 遭到拒绝的原因，请在运行回收程序时加上 -d 选项。它将显示有关回收程序如何选择 VSN 进行回收的信息。

第八章

升级环境中的硬件

本章介绍如何升级现有 Sun StorEdge SAM-FS 环境中的硬件。它包括下列主题：

- 第 174 页的 “在自动化库中添加端口”
- 第 175 页的 “升级或更换库”
- 第 178 页的 “升级 DLT 磁带驱动器”

此外，还需在 Sun StorEdge SAM-FS 环境中执行某些其他类型的操作以及升级。下列出版物对这些操作进行了介绍：

- *《Sun StorEdge QFS 和 Sun StorEdge SAM-FS 软件安装和配置指南》*介绍了升级过程 Sun StorEdge QFS 和 Sun StorEdge SAM-FS 软件。
- *《Sun StorEdge QFS 和 Sun StorEdge SAM-FS 文件系统管理指南》*介绍了以下几类操作和升级过程：
  - 初始化文件系统
  - 将配置文件的更改应用至系统中
  - 安装文件系统
  - 卸载文件系统
  - 检查文件系统的完整性和修复文件系统
  - 保存升级信息
  - 为硬件设备升级做准备
  - 添加磁盘高速缓存到文件系统
  - 更换文件系统中的磁盘
  - 升级主机系统
  - 升级 Solaris 操作系统

在自动化库中添加端口

软件许可证控制着 Sun StorEdge SAM-FS 系统可管理的卡盒端口的数量。要增加端口的数量，请执行本节所述步骤。

▼ 在库中添加端口

1. **确定您是否需要通过授权服务供应商 (ASP) 获取一组新的许可证密钥，如果您的帐户未分配 ASP，请通过 Sun Microsystems 获取许可证密钥。**

   如果不需要新的许可证密钥，请继续执行步骤 2。

   如果需要新的许可证密钥，请执行下列步骤：

   a. **使用新的许可证密钥取代现有的许可证密钥。**

      如果您已获得新的许可证密钥，请执行本步骤。许可证密钥从下面文件的第一列开始：

      ```
      /etc/opt/SUNWsamfs/LICENSE.4.1
      ```

      其中不可包含其它任何关键字、主机 ID 或其它信息。

   b. **运行 `samd`(1M) `config` 命令，以使 Sun StorEdge SAM-FS 软件可识别新的许可证密钥。（可选）**

      如果您已获得新的许可证密钥，请执行本步骤。例如：

      ```
      # samd config
      ```

2. **卸载库目录。**

   按以下格式使用 `sacmd`(1M) `unload` 命令：

   ```
   samcmd unload eq
   ```

   其中的 *eq*，用于指定自动化库在 `mcf` 文件中定义的设备序号。此命令可将库目录条目移入历史记录目录，并保存每一个卡盒的目录信息。

   执行此命令后，可使用 `samu`(1M) `v` 显示选项进行检查，您将发现自动化库的 `v` 显示屏幕为空，而历史记录的 `v` 显示屏幕则显示曾位于自动化库中的 VSN。

3. 关闭 Sun StorEdge SAM-FS 系统。

   有关如何执行本步骤的信息，请参阅第 13 页的 “在 Sun StorEdge SAM-FS 环境中使用自动化库和手动载入的驱动器”。

4. 根据制造商建议的过程关闭主机系统和库的电源。

5. 请库硬件工程师将端口添加至自动化库。

6. 使用标准启动过程打开主机系统的电源。

7. 启动 Sun StorEdge SAM-FS 系统。

   有关如何执行本步骤的信息，请参阅第 13 页的 “在 Sun StorEdge SAM-FS 环境中使用自动化库和手动载入的驱动器”。新的许可证信息将出现在 samu(1M) 实用程序的 l 显示屏幕中。

升级或更换库

断开连接并安装另一自动化库之前，请根据 *《Sun StorEdge QFS 和 Sun StorEdge SAM-FS 文件系统管理指南》*中的 “准备升级硬件”部分所述，为升级做好准备。

▼ 更换或升级库

1. 卸载库目录。

   按以下格式使用 sacmd(1M) unload 命令：

```
samcmd unload eq
```

   其中的 *eq*，用于指定自动化库在 mcf 文件中定义的设备序号。此命令可将库目录条目移入历史记录目录，并保存每一个卡盒的目录信息。

   执行此命令后，可使用 samu(1M) v 显示选项进行检查，您将发现自动化库的 v 显示屏幕为空，而历史记录的 v 显示屏幕中则显示曾位于自动化库中的 VSN。

2. **更新** `/etc/opt/SUNWsamfs/inquiry.conf` **文件。（可选）**

   新库应在此文件中按供应商、自动化库型号和 Sun StorEdge SAM-FS 内部名标识出来。

   例如，释放的 `inquiry.conf` 文件中包含下面的行：

```
"HP",    "C1710T",  "hpoplib"    # HP optical library
```

   该行表示如果系统检测到由 `HP` 制造的型号为 `C1710T` 的 SCSI 设备，则系统将它作为 `hpoplib` 进行驱动。前两个字段（供应商 / 产品）由硬件设备得出。最后一个字段 `hpoplib` 是系统内部使用的名称，系统使用该名称来确定如何与该设备进行通信。如果 `inquiry.conf` 文件需要更改，则所做的更改只有在重新启动 `sam-amld` 后台程序之后才能生效。

3. **将当前** `/etc/vfstab` **文件另存为** `/etc/vfstab.cur`。

4. **编辑** `/etc/vfstab` **文件**。

   将所有 Sun StorEdge SAM-FS 安装选项从 `yes` 更改为 `no`。

5. **将** `/etc/opt/SUNWsamfs/archiver.cmd` **文件另存为** `archiver.cmd.cur`。

6. **编辑** `/etc/opt/SUNWsamfs/archiver.cmd` **文件**。

   将 `wait` 指令添加在第一行。

7. **根据制造商建议的过程关闭主机系统和外围设备的电源。**

8. **断开自动化库的连接。**

9. **将连接电缆连接至新的自动化库。**

10. **根据制造商建议的开机顺序打开外围设备和主机系统的电源。**

11. **确保主机系统可以识别新的自动化库。**

    输入下面的命令：

```
> probe-scsi-all
```

    继续下一步骤之前，新的自动化库及其驱动器必须显示在该命令的输出结果中。如果系统不能识别自动化库及其驱动器，则它们可能存在连接问题。

12. **引导系统。**

    输入下面的命令以在新配置下引导系统：

```
> boot -rv
```

13. **如果驱动器或自动化库的目标号发生了变化，或者驱动器在自动化库中的顺序或编号发生了变化，请修改 `/etc/opt/SUNWsamfs/mcf` 文件以反映出新的配置信息。（可选）**

    这类似于 *《Sun StorEdge QFS 和 Sun StorEdge SAM-FS 软件安装和配置指南》*中所述的初始安装过程。

14. **创建新的 `/dev/samst` 条目。（可选）**

    如果您添加了新设备，请执行本步骤。输入下面的命令：

```
# samdev
```

15. **初始化 Sun StorEdge SAM-FS 系统。**

    您可以通过安装文件系统或输入下面的命令来执行本步骤。

```
# samd start
```

    系统在初始化期间会检测到自动化库中的端口数量发生了变化。系统将对自动化库运行全面的核查以更新库目录。恢复存档之前，必须完成全面核查过程。

    如果在核查期间出现问题，则很可能是因为驱动器在自动化库中的顺序与其在 `/etc/opt/SUNWsamfs/mcf` 文件中的顺序不符。记住，驱动器有两个属性：SCSI 目标 ID，以及它在自动库中的位置。无论是在升级之前还是在升级之后，这两个属性均必须正确无误。

    如果核查顺利完成，请继续进行下一步骤。

16. **使用升级之前的版本替换 `/etc/vfstab` 和 `/etc/opt/SUNWsamfs/archiver.cmd` 文件。**

    分别使用已保存的 `/etc/vfstab.cur` 和 `/etc/opt/SUNWsamfs/archiver.cur` 文件。

17. **重新引导系统，确保配置不存在任何错误。**

自动化库按位置编号调用驱动器。例如，当系统希望将卡盒载入驱动器时，它必须向自动化库发出命令以将卡盒从端口 123 载入至驱动器 3。

根据第三个 `mcf` 条目，驱动器 3 可能是 SCSI 目标 6。系统之所以知道它是驱动器 3，是因为它是 `mcf` 文件中的第三个驱动器条目。自动化库之所以知道它是驱动器 3，是根据它在自动化库中占据的物理位置。

请求自动化库将卡盒载入驱动器之后，系统将测试驱动器是否处于设备就绪状态。此时，系统将使用 `mcf` 文件中 `/dev/samst/`*`scsi-target`* 条目定义的 SCSI 目标 ID。因此，这些条目与刚才载入卡盒的驱动器相符是非常重要的。

目前，没有好的方法来确定此类信息。通常，制造商在自动化库出厂时为驱动器设置了升序的 SCSI ID，但这一点并没有保证。确定这一点的方法之一就是，使用 `samu`(1M) 实用程序的 `:load` 命令载入一个卡盒，然后观察 `samu`(1M) 实用程序的 `s` 显示屏幕，以确定哪个驱动器的 `t` 显示屏幕状态标记为 `r` 而不是 `p`。

升级 DLT 磁带驱动器

为了利用拥有更高密度和更快速度的磁带技术，可以升级自动化库中的 DLT 磁带驱动器或独立磁带驱动器。例如，您可以将 DLT 4000 驱动器升级至 DLT 7000 驱动器。

要在 Sun StorEdge SAM-FS 环境中进行升级，应该在启动 Sun StorEdge SAM-FS 环境之前，添加新的驱动器，重新引导新的配置，并根据需要更新 `mcf` 文件。此外，如果您要升级端口的数量，则还需联系授权服务供应商 (ASP) 或 Sun Microsystems，因为您可能需要升级许可证。

升级驱动器之前，请注意以下限制和常规信息：

- Sun StorEdge SAM-FS 环境中不支持在同一个直接连接的自动化库中混用不同型号的 DLT 磁带驱动器。例如， Sun StorEdge SAM-FS 系统不能区分同一个自动化库中的 DLT 4000 磁带驱动器和 DLT 7000 磁带驱动器。因此，您必须同时使用新的驱动器更换所有的旧 DLT 驱动器。
- 低密度磁带不能与高密度磁带或磁带驱动器混合使用。不过，您可以继续使用高密度驱动器来读取低容量的磁带或向其中写入数据。
- 为了充分利用高密度 DLT 磁带，您可能希望回收现有的文件并将它们移植到高密度磁带中。要执行此操作，请将所有低密度磁带标记为只读，然后标记要回收的磁带。有关回收磁带的信息，请参阅第 159 页的 “回收”。
- 当标记每一个磁带时，系统将确定磁带的密度并将其记录至库目录中。

▼ 升级磁带驱动器

1. **确定您的当前转储文件是否完整无缺。（可选）**

   如果不是，请在继续下一步骤之前对文件系统执行 `samfsdump`(1M)。

2. **更新 `/kernel/drv/st.conf` 文件以标识新的驱动器。**

   在此文件中，磁带驱动器是按其供应商、磁带型号和 Sun StorEdge SAM-FS 内部名称进行标识的。例如，已释放的 `st.conf` 文件中包含下面的行：

   ```
   "QUANTUM DLT7000", "DLT 7000 tape drive",   "dlt7-tape"
   ```

   `/opt/SUNWsamfs/examples/st.conf_changes` 中提供了文件示例。您可以将整个文件读入 `/kernel/drv/st.conf`，也可合并所需的更改。有关更新 `st.conf` 文件的详细信息，请参阅 *《Sun StorEdge QFS 和 Sun StorEdge SAM-FS 软件安装和配置指南》*。

3. **根据制造商建议的过程关闭主机系统和外围设备的电源。**

4. **使用新的磁带驱动器更换旧的磁带驱动器。**

5. **根据制造商建议的开机顺序打开外围设备和主机系统的电源。**

6. **确保主机系统可以识别新的驱动器。**

   输入下面的命令：

   ```
   > probe-scsi-all
   ```

   继续下一步骤之前，自动化库及新驱动器必须显示在该命令的输出结果中。如果未显示这些驱动器，则可能存在连接问题，请予以更正。该命令返回预期的信息之后，您可以继续进行下一步。

7. **引导系统。**

   输入下面的命令以在新配置下引导系统：

   ```
   > boot -rv
   ```

8. **修改 `/etc/opt/SUNWsamfs/mcf` 文件以反映新的配置信息。（可选）**

   如果驱动器或自动化库的目标号发生了变化，或者驱动器在自动化库中的顺序或编号发生了变化，请执行本步骤。这类似于 *《Sun StorEdge QFS 和 Sun StorEdge SAM-FS 软件安装和配置指南》*中所述的初始安装过程。

9. **为新设备创建新的 `/dev/samst` 条目。（可选）**

   要创建这些条目，请输入下面的命令：

```
# samdev
```

10. **启动 Sun StorEdge SAM-FS 系统。**

11. **安装文件系统。**

    您现在可以继续使用现有的 Sun StorEdge SAM-FS 磁带了。

第九章

高级内容

本章介绍超出基本系统管理和使用范围的高级内容。

它包括下列主题：

设备日志

设备日志工具可以提供设备专用的错误信息，您可以使用这些信息来分析某些类型的设备问题。它可帮助您确定自动化库、磁带驱动器或光盘驱动器发生故障事件的顺序。请注意，设备日志工具并不能收集软介质错误，如可恢复的读取错误。

系统将设备日志消息分别写入至各个不同的日志文件。每一个自动化库、磁带和光盘驱动器设备均有一个日志文件，并且历史记录也有一个日志文件。日志文件位于 `/var/opt/SUNWsamfs/devlog` 目录下。每一个日志文件的名称分别与相应设备的设备序号相同。

示例。假设您的环境中具有 Sun StorEdge SAM-FS 文件系统和一个配有两个光盘驱动器的 Hewlett Packard 光盘库。

代码示例 9-1 显示了 mcf 文件。

代码示例 9-1 mcf 文件示例

```
/dev/samst/c1t5u0   40   hp   hp40   -   etc/opt/SUNWsamfs/hp40_cat
/dev/samst/c1t4u0   41   mo   hp40   -
/dev/samst/c1t6u0   42   mo   hp40   -
```

代码示例 9-2 显示了 /var/opt/SUNWsamfs/devlog 文件。

代码示例 9-2 devlog 文件

```
# pwd
/var/opt/SUNWsamfs/devlog
# ls
40       41       42       43
#
```

设备 43 是历史记录。

何时使用设备日志

设备日志可以方便地生成许多日志消息，特别适用于已打开所有设备的全部日志选项且具有大量设备活动的场合。设备日志设置最初设置为下面的默认值：

```
err retry syserr date
```

如果您怀疑 Sun StorEdge SAM-FS 环境中配置的某个设备出现问题，则应为该设备启用额外的日志事件。另外，如果您的服务供应商建议您这样做，也应启用设备日志。在这些情况下，请将事件设置为 detail（详细）。在极为特殊的情况下，您的服务供应商可能建议您将某个设备的事件设置为 all（全部）。这可以添加额外的日志信息。不过，一般而言，在系统运行时设置过多的日志并无益处，甚至是不可行的。

当运行 samexplorer(1M) 命令时，系统会自动收集设备日志信息。这样，作为问题分析活动的一部分，文件系统服务人员可以复查任何可能的设备错误信息。

启用设备日志

您可以使用两种方法来启用设备日志。

对于第 1 种和第 2 种方法：

- *eq* 是设备在 mcf 文件中定义的设备序号，或对于所有设备，它为关键字 all。
- samset(1M) 手册页中列出了设备日志事件。另外，第 182 页的 “启用设备日志”也列出了这些设备日志事件。请注意，设备日志消息仅以英文文本的格式提供。 *event* 是下表中的一种或多种事件类型：
  - all
  - date
  - default
  - detail
  - err
  - event
  - label
  - mig
  - module
  - msg
  - none
  - retry
  - stage
  - stage_ck
  - syserr
  - tapealert
  - time

可按以下两种方式中的一种启用设备日志。这些过程如下所示：

- 第 183 页的 “使用 samset(1M) 命令启用设备日志”
- 第 184 页的 “通过编辑 defaults.conf 文件启用设备日志”

▼ 使用 samset(1M) 命令启用设备日志

● **使用 samset(1M) 命令。**

例如：

```
# samset devlog eq event
```

有关 samset(1M) 命令的详细信息，请参阅 samset(1M) 手册页。

▼ 通过编辑 defaults.conf 文件启用设备日志

1. 成为超级用户。
2. 使用 vi(1) 或其它编辑器打开文件 /etc/opt/SUNWsamfs/defaults.conf。
3. 在 defaults.conf 文件中添加 devlog 指令。

   添加以下指令：

   ```
   devlog eq event
   ```

   其中的 *eq*，用于指定要为其记录消息的设备的设备序号。

   而其中的 *event*，用于指定一个或多个如第 182 页的 “启用设备日志”中所述的事件。如果要指定多个事件，请用空格进行分隔。

   Sun StorEdge SAM-FS 文件系统会在启动时，自动将每个可用设备的事件类型设置为 default。另外，还可以使用 samset(1M) 命令来确定每一个设备日志的当前设置。
4. 保存并关闭 defaults.conf 文件。
5. 使用 samd(1M) config 命令应用 defaults.conf 文件中的更改。

   ```
   # samd config
   ```

可移动介质文件

您可以使用 request(1) 命令手动创建、写入和读取那些不使用磁盘高速缓存来缓冲数据的文件。采用这种方式创建的文件称为 *可移动介质文件*。

与典型的 Sun StorEdge SAM-FS 文件类似，可移动介质文件也具有权限、用户名、组名和大小属性。但是，这些文件的数据并不保留在磁盘高速缓存中。因此，您可以创建文件大小大于磁盘高速缓存空间的文件，并将它们写入可移动介质卡盒。对于您在 request(1) 命令中指定的文件，系统会为它在 .inodes 文件中创建索引节点条目。用户并不需要知道文件在可移动介质中的起始点。（对于那些数据位于磁盘高速缓存中的文件来说，情形与此相同。） Sun StorEdge SAM-FS 文件系统会从索引节点条目中读取信息。多个可移动介质文件可以保留在同一个卷中。

如果可移动介质文件存储在多个卷中，那么将该文件称为卷溢出文件。*卷溢出* 功能允许系统在多个卡盒的多个卷中存储单个文件。卷溢出文件是一种可移动介质文件。如果您有非常大的文件，文件大小超出了所选介质的容量，那么卷溢出功能将非常有用。

▼ 创建可移动介质或卷溢出文件

1. **使用 `tplabel`(1M) 或 `odlabel`(1M) 命令标记磁带或磁光盘卡盒。**
   有关这些命令的详细信息，请参阅各自的手册页。

2. **使用 `request`(1M) 命令。**
   而且该命令至少应该使用以下选项：

```
request -m media_type -v vsn [vsn/vsn ...][-l vsn_file] input_file
```

表 9-1 `request`(1) 命令的变量

变量	含义
media_type	可移动介质卡盒的介质类型。有关有效 *media_type* 参数的详细信息，请参阅 `mcf`(4) 手册页。
vsn	可移动介质卡盒的卷序列名。 如果指定了多个 *vsn*，那么将创建卷溢出文件。您最多可以为卷溢出文件指定 256 个 *vsn*。应该使用正斜杠字符 (/) 来分隔 *vsn* 变量。所指定的 *vsn* 不应该是 Sun StorEdge SAM-FS 环境中用于自动存档的卷。每次存档时，系统均会将下一个要存档的文件添加至当前数据的末尾，并将 EOF 标签移至数据的后面。
vsn_file	包含 *vsn* 列表的输入文件。如果您拥有许多 *vsn*，那么在输入文件中指定 *vsn* 列表要比在命令行中便捷得多。
input_file	要写入至可移动介质卡盒的文件。该文件必须位于 Sun StorEdge SAM-FS 文件系统中。

示例 1。以下命令用于创建可移动介质文件。

```
# request -m lt -v aaa rem1
```

示例 2。以下命令用于在三个卷中创建卷溢出文件：

```
# request -m lt -v TAPE01/TAPE02/TAPE03 large.file
```

您必须依次读取和写入可移动介质文件。如果请求的卷位于 mcf 文件所定义的自动化库中，那么 Sun StorEdge SAM-FS 文件系统会自动安装该卷。

如果卷中存在可移动介质文件，则回收程序不能对该卷进行回收。回收程序希望只有已存档的文件位于指定用于存档的特定卷中。另外，存档程序永远不会存档可移动介质文件。

NFS 不支持可移动介质文件。

请注意，使用 request(1) 命令将会忽略存档程序的一般功能。

有关说明如何创建可移动介质文件的示例，请参阅 request(1) 手册页。

分段文件

Sun StorEdge SAM-FS 环境支持分段文件。将文件分成多个段可以提高磁带存储检索速度、改善存取性能以及增强大文件的可管理性。分段文件的大小可以大于物理磁盘高速缓存。使用分段文件，您可以只将文件的一部分时刻保留在磁盘高速缓存中。

可以使用 segment(1) 命令来指定分段大小。您所设置的分段大小不能小于当前文件的大小。

分段文件支持磁带拆分功能。将文件分段后，可将文件同时拆分至多个磁带设备中，这样大大缩短了存储各个文件段的时间。由于用户只需恢复所需的文件段（而不是整个文件），因此提高了数据访问速度。

由于只有发生更改的文件部分才需要重新存档，因此分段还可以提高存档效率。文件的各个分段可以并行存档，并且分段文件可以并行登台。这提高了系统的存档和恢复性能。

用户可以为文件、目录或整个文件系统启用分段功能。分段文件支持其它所有 Sun StorEdge SAM-FS 功能。

以下几节说明了分段文件与非分段文件之间的差异。有关分段文件的详细信息，请参阅 segment(1) 或 sam_segment(3) 手册页。

存档

对于分段文件，存档单位是文件段自身，而不是整个文件。所有存档属性和优先级均应用于单个文件段，而不应用于整个文件。

所存档的单位是文件段。您可以通过在 archiver.cmd 文件中为存档组指定 -drives 和 -drivemin 参数，来拆分段。

例如，假定文件系统中有一个大小为 100 MB 的分段文件，其段大小为 10 MB。如果 archiver.cmd 中使用 -drives 2 指令定义存档组，则该分段文件将并行存档至 2 个驱动器。段 1、3、5、7 和 9 存档在第一个驱动器中，而段 2、4、6、8 和 10 存档在第二个驱动器中。

存档程序只存档已发生更改的文件段 — 而不是整个文件。最多可以为每一个文件段创建四个存档副本。 Sun StorEdge SAM-FS 支持对文件段使用卷溢出功能。

注意 – 分段文件的索引不含用户数据。索引被视为元数据，由系统分配至文件系统存档组。

故障恢复

有关在出现故障时恢复分段文件的详细信息，请参阅 *《Sun QFS、 Sun SAM-FS 和 Sun SAM-QFS 故障恢复指南》*。

系统错误工具报告

系统错误工具 (SEF) 报告系统用于收集自动化库中的磁带设备生成的日志检测数据，然后将这些数据写入至日志文件并转换成可读的格式。它包括以下项目：

- 日志文件，包含从磁带设备日志检测页收集的数据。
- sefreport(1M) 命令，以可读的格式将日志文件写入至 stdout。该日志文件可以作为用户分析脚本的输入项。

sefreport(1M) 命令可读取 Sun StorEdge SAM-FS SEF 日志文件的内容。日志文件中包含了从设备的日志检测页收集到的数据，这些设备是 Sun StorEdge SAM-FS 环境中所使用的外部磁带设备。日志检测页因供应商而异。有关参数代码、控制位和参数值的含义，请参阅每一个特定设备的供应商文档。

独立磁带驱动器不支持 SEF。对于那些不支持 tapealert(1M) 功能的旧 SCSI-2 设备， SEF 报告功能非常有用。有关详细信息，请参阅 tapealert(1M) 手册页。

▼ 启用 SEF 报告

1. 成为超级用户。
2. 使用 mkdir(1) 命令创建 SEF 目录。

   例如：

```
# mkdir /var/opt/SUNWsamfs/sef
```

3. 使用 touch(1) 命令启用 SEF 报告功能。

   在安装报告系统后，您随时可以通过创建 sefdata 日志文件来启用 SEF 报告功能。起初， SEF 日志文件必须为空。

```
# touch /var/opt/SUNWsamfs/sef/sefdata
```

   上面的命令示例表明所创建的 SEF 日志文件位于 /var/opt/SUNWsamfs/sef/sefdata 目录中。这是默认位置。

4. 使用 samd(1M) stop 和 samd(1M) start 初始化 SEF 报告功能。

   例如：

```
# samd stop
# samd start
```

   生成的 SEF 数据将添加至日志文件的末尾。

   您可以对 SEF 报告功能进行配置，以便在其它位置记录和读取日志检测数据。有关从其它位置读取日志检测文件的详细信息，请参阅 sefreport(1M) 手册页。

SEF 报告输出

使用 sefreport(1M) 命令之前，请确保 /opt/SUNWsamfs/sbin 位于您的命令路径中。 SEF 报告输出的内容由标题行和日志检测数据组成。

记录中每一页的日志检测数据将打印在标题行的后面。对于每一日志检测数据页，系统均会打印用于标识页码的行，随后是一行列标题。接下来打印数据，每行三列，各列的标题分别如下：param code、control 和 param value。所有生成的数据均采用十六进制格式。

▼ 生成 SEF 输出

- **使用 `sefreport`(1M) 命令生成 SEF 输出。**

  下面是 `sefreport`(1M) 命令的最常用选项：

  - `-d` 选项。 `-d` 选项用于生成附加的设备信息。它将为每一条记录写入附加的包含设备序号和路径名的标题行。这便于用户搜索和查找与特定设备相关的 SEF 记录。
  - `-v` 或 `-t` 选项。

    `-v` 选项用于生成详细的信息。它将与设备序号、页码和 VSN 相关的信息添加至记录的每一行。这使用户可以只选择与特定设备或特定卷相关的行。

    `-t` 用于生成包含文字说明的日志检测页。对于日志检测页输出的每一行，报告中都包含了一个额外的字符串，字符串的内容包括：设备序号、页码、 VSN 和参数编码说明。

    不要在同一命令行中指定 `-t` 和 `-v` 选项。它们是互相排斥的。

  例如，下面的 SEF 命令从默认位置读取 SEF 日志文件，写入每一个设备的设备编号和路径名，并生成输出：

```
# sefreport -d /var/opt/SUNWsamfs/sef/sefdata > sef.output
```

代码示例 9-3 显示了 `sef.output` 文件的内容。

代码示例 9-3 `sef.output` 的内容

```
Record no. 1
Mon Mar 26 11:17:48 2001  STK      9840            1.25 VSN 002981
   Eq no. 32   Dev name /dev/rmt/1cbn

   PAGE CODE 2
   param code  control   param value
        00h       74h      0x0
        01h       74h      0x0
        02h       74h      0x0
        03h       74h      0x0
        04h       74h      0x0
        05h       74h      0x40050
        06h       74h      0x0

   PAGE CODE 3
   param code  control   param value
        00h       74h      0x0
        01h       74h      0x0
```

代码示例 9-3 sef.output 的内容 *（接上页）*

```
        02h        74h      0x0
        03h        74h      0x0
        04h        74h      0x0
        05h        74h      0x140
        06h        74h      0x0

    PAGE CODE 6
    param code  control   param value
        00h        74h      0x0

Record no. 2
Mon Mar 26 11:30:06 2001  STK      9840            1.25 VSN 002999
    Eq no. 31   Dev name /dev/rmt/0cbn

    PAGE CODE 2
    param code  control   param value
        00h        74h      0x0
        01h        74h      0x0
        02h        74h      0x0
        03h        74h      0x0
        04h        74h      0x0
        05h        74h      0x1400a0
        06h        74h      0x0

    PAGE CODE 3
    param code  control   param value
        00h        74h      0x0
        01h        74h      0x0
        02h        74h      0x0

        03h        74h      0x0
        04h        74h      0x0
        05h        74h      0x190
        06h        74h      0x0

    PAGE CODE 6
    param code  control   param value
        00h        74h      0x0

Record no. 3
Mon Mar 26 11:30:23 2001  STK      9840            1.25 VSN 002981
    Eq no. 32   Dev name /dev/rmt/1cbn
```

代码示例 9-3 sef.output 的内容 *（接上页）*

```
    PAGE CODE 2
    param code  control   param value
         00h        74h      0x0
         01h        74h      0x0
         02h        74h      0x0
         03h        74h      0x0
         04h        74h      0x0
         05h        74h      0x18400f0
         06h        74h      0x0


    PAGE CODE 3
    param code  control   param value
         00h        74h      0x0
         01h        74h      0x0
         02h        74h      0x0
         03h        74h      0x0
         04h        74h      0x0
         05h        74h      0x1e0
         06h        74h      0x0

    PAGE CODE 6
    param code  control   param value
         00h        74h      0x0
.
.
.
```

注意 – 受本手册的内容所限，上面的输出已作了删节。

有关 SEF 日志文件的详细信息（包括内容和格式），请参阅 sefdata(4) 手册页。有关可选 SEF 报告格式的详细信息，请参阅 sefreport(1M) 手册页。

管理 SEF 日志文件

SEF 日志文件的管理方式，与任何其它 Sun StorEdge SAM-FS 日志文件相同。用户可以定期运行 cron(1) 作业，以将当前日志文件保存至另一位置、删除旧 SEF 文件、创建新 （空）的 SEF 文件或执行其它任务。

另外，用户还可使用 log_rotate.sh(1M) 实用程序来循环更新该日志文件。

有关用于管理 SEF 日志文件的工具的详细信息，请参阅 cron(1) 或 log_rotate.sh(1M) 手册页。

词汇表

A

安装点 (mount point) 安装文件系统的目录。

B

备份存储 (backup storage) 一组文件的快照，旨在防止意外丢失数据。备份不仅包括文件的属性，而且还包括关联的数据。

本地文件系统 安装在 Sun Cluster 的某一个节点上的文件系统，它对于其他节点来说，可用性不高。也可以指安装在独立服务器上的文件系统。

C

拆分 (striping) 一种以交叉方式将文件同时写入至多个逻辑磁盘的数据存取方法。所有 Sun StorEdge QFS 和 Sun StorEdge SAM-FS 文件系统都允许您声明，对每个单独的文件系统进行拆分式或循环式存取。 Sun StorEdge QFS 文件系统允许您在每个文件系统中声明拆分组。另请参阅 “循环”条目。

拆分大小 (stripe size) 移至拆分的下一个设备之前，要分配的磁盘分配单元 (DAU) 的数量。如果 `stripe=0`，则文件系统采用循环存取方式，而不采用拆分存取方式。

拆分组 (striped group)
Sun StorEdge QFS 文件系统中的一组设备，在 `mcf` 文件中定义为一个（通常为两个）或多个 g*XXX* 设备。拆分组作为一个逻辑设备使用，并且始终拆分成若干个大小等于磁盘分配单元 (DAU) 的空间。一个文件系统中可以指定多达 128 个拆分组，但在整个系统中，最多不能超过 252 个设备。

超级块 (superblock)
文件系统中的一种数据结构，用于定义文件系统的基本参数。它由系统写入到存储设备系列集中的所有分区，以识别该系列集中的各个分区成员。

传输器 (robot)
自动化库的一部分，用于在存储端口和驱动器之间移动卡盒。也称传输设备。

磁盘拆分 (disk striping)
指在数个磁盘上记录同一个文件的过程，因此可以提高存取性能和增加整体存储容量。另请参阅 “拆分”条目。

磁盘分配单元 (disk allocation unit)
参阅 DAU。

磁盘高速缓存 (disk cache)
Sun StorEdge SAM-FS 文件系统软件的常驻磁盘部分。它用于在在线磁盘高速缓存与存档介质之间创建和管理数据文件。单个磁盘分区或整个磁盘均可用作磁盘高速缓存。

磁盘缓冲区 (disk buffer)
使用 Sun SAM-Remote 软件时，磁盘缓冲区是指在将数据从客户端存档至服务器时所用的服务器系统上的缓冲区。

磁盘空间阈值 (disk space threshold)
管理员定义的供用户使用的磁盘空间数量。此项功能用于定义理想磁盘高速缓存利用率的范围。阈值上限表示磁盘高速缓存利用率的最大级别。阈值下限表示磁盘高速缓存利用率的最小级别。释放程序依据这些预定义的磁盘高速缓存空间阈值来控制磁盘高速缓存的利用率。

存储端口 (storage slot)
自动化库中的存储位置。卡盒不在驱动器中使用时，将会存储在存储端口内。如果是直接连接的库，则存储端口的内容保存在自动化库的目录中。

存储系列集 (storage family set)
由一系列磁盘组成，整体表现为单个磁盘系列设备。

存档程序 (archiver)
一种可以自动将文件复制到可移动卡盒的存档软件程序。

存档存储 (archive storage)
已在存档介质中创建的文件数据副本。

存档介质 (archive media)
存档文件所写入的介质。存档介质可以是库中的可移动磁带或磁光盘卡盒。此外，它还可以是另一系统中的安装点。

D

DAU （磁盘分配单元， Disk allocation unit）最基本的在线存储单元。也称 “块大小”。

此外， Sun StorEdge QFS 文件系统还支持大小完全可调的 DAU，范围从 16 KB 到 65,528 KB 不等。不过，所指定的 DAU 必须是 8 KB 的倍数。

Sun StorEdge SAM-FS 文件系统既支持小 DAU，也支持大 DAU。小 DAU 是指 4 KB （2^{14} 或 4096 字节）。大 DAU 是指 16、 32 或 64 KB。可用的 DAU 大小配对包括 4/16、 4/32 和 4/64。

登台 (staging) 是指将近线或离线文件从存档存储设备恢复至在线存储设备的过程。

多阅读器文件系统 (Multireader file system) Sun StorEdge QFS 多阅读器文件系统是指允许单写入者、多阅读者的文件系统，这种功能使您可以指定能够安装在多台主机上的文件系统。多台主机可以读取该文件系统，但只有一台主机可以向该文件系统写入数据。多阅读者主机可通过 `mount`(1M) 命令的 `-o reader` 选项指定。而单写入者主机可通过 `mount`(1M) 命令的 `-o writer` 选项指定。有关 `mount`(1M) 命令的详细信息，请参阅 `mount_samfs`(1M) 手册页。

F

Family Set （系列集） 由一组独立物理设备组成的存储设备，例如自动化库中的磁盘组或驱动器组。另请参阅 “系列集”。

FDDI （光纤分布式数据接口， Fiber distributed data interface）一种传输速度为 100 MB/s 的光纤局域网。

FTP （文件传输协议， File transfer protocol）一种通过 TCP/IP 网络在两台主机之间传送文件的网际协议。

范围阵列 (extent array) 文件的索引节点中的阵列，用于定义分配给文件的每个数据块在磁盘上的位置。

分区 (partition) 设备的一部分或磁光盘卡盒的一面。

G

光纤分布式数据接口 (fibre-distributed data interface) 参阅 FDDI。

光纤信道 (fibre channel) 由 ANSI 提出的标准，规定在设备之间实行高速串行通信。光纤信道是 SCSI-3 中的一种总线结构。

H

核查 (audit)（全面） 载入卡盒并验证其 VSN 的过程。对于磁光盘卡盒，此过程会确定其容量和空间信息，然后将这些信息输入到自动化库的目录中。

回收程序 (recycler) 一个 Sun StorEdge SAM-FS 实用程序，用于收回由过期存档副本占用的卡盒空间。

J

计时器 (timer) 一种限额软件，用于跟踪用户已在为其设定的软限制和硬限制之间经历的时间。

间接块 (indirect block) 包含存储块列表的磁盘块。 Sun StorEdge QFS 和 Sun StorEdge SAM-FS 文件系统中最多可有三级间接块。第一级间接块包含数据存储块的列表。第二级间接块包含第一级间接块的列表。第三级间接块包含第二级间接块的列表。

介质 (media) 磁带或光盘卡盒。

介质回收 (media recycling) 回收或再利用低利用率存档介质的过程。低利用率存档介质是指所包含的活动文件非常少的存档介质。

近线存储设备 (nearline storage) 一种可移动介质存储设备，访问此类设备之前需要启用自动安装功能。近线存储设备通常比在线存储设备便宜，但所需的访问时间相对长一些。

镜像写入 (mirror writing)
在互不相连的磁盘组中保存两份副本的过程，用于防止因单个磁盘损坏而导致数据丢失。

卷 (volume)
卡盒中用于共享数据的命名区域。一个卡盒中可以有一个或多个卷。双面卡盒有两个卷，每一面为一个卷。

卷溢出 (volume overflow)
一种允许系统在多个卷上存储单个文件的功能。对于使用非常大型文件 （超过单个卷的容量）的站点，卷溢出功能非常有用。

K

卡盒 (cartridge)
一种包含数据记录介质的物品。例如，磁带或光盘。有时称为*介质*、*卷*或*媒体*。

可寻址存储设备 (addressable storage)
包括在线、近线、离站和离线存储的存储空间，用户可通过 Sun StorEdge QFS 或 Sun StorEdge SAM-FS 文件系统访问这些空间。

可移动介质文件 (removable media file)
一种特殊类型的用户文件，可以直接从它所在的可移动介质卡盒 （如磁带或光盘卡盒）中进行访问。也可用于写入存档和登台文件数据。

客户端 — 服务器
分布式系统中的交互模式，在此模式下，一个站点上的程序向另一站点上的程序发送请求并等待回应。发送请求的程序称为 “客户端”。提供回应的程序称为 “服务器”。

库 (library)
参阅 “自动化库”。

库目录 (library catalog)
参阅 “目录”。

块大小 (block size)
参阅 DAU。

块分配图 (block allocation map)
一个显示磁盘中所有可用存储块的位图，它指出了每个块的状态，即是在使用中还是未使用。

宽限期 (grace period)
对于磁盘限额而言，是指用户在达到为其设定的软限制之后，系统允许用户继续创建文件和分配存储空间的时间期限。

L

LAN 局域网 (Local area network)。

LUN 逻辑单元编号 (Logical unit number)。

离线存储设备 (offline storage) 需要操作员参与才能载入的存储设备。

离站存储设备 (offsite storage) 远离服务器的用于故障恢复的存储设备。

连接 (connection) 两个协议模块之间的通道，用于提供稳定可靠的数据流传输服务。 TCP 连接可以从一台计算机上的 TCP 模块扩展到另一台计算机上的 TCP 模块。

M

mcf 主配置文件 (master configuration file)。初始化期间读取的文件，用于定义 Sun StorEdge QFS 或 Sun StorEdge SAM-FS 环境内部各个设备 （拓扑结构）之间的关系。

命名空间 (name space) 一组文件的元数据部分，用于标识文件及其属性和存储位置。

目录 (catalog) 自动化库中 VSN 的记录。每个自动化库均有一个目录，并且一个站点拥有一个历史记录，用于记录所有自动化库。

目录 (directory) 指向文件系统中其它文件和目录的文件数据结构。

N

NFS 网络文件系统 (Network file system)。一种 Sun 分布式文件系统，可以对异构网络上的远程文件系统进行透明访问。

NIS SunOS 4.0 （最小）网络信息服务 (Network Information Service)。一种分布式网络数据库，包含与网络中系统和用户有关的关键信息。 NIS 数据库存储在主服务器和所有从属服务器上。

内核 (kernel) 提供基本系统功能的中央控制程序。 UNIX 内核可以创建和管理进程；提供存取文件系统的功能；提供基本安全性能；以及提供通信功能。

P

排列预备请求的优先顺序 (prioritizing preview request) 为不能立即得到满足的存档和登台请求分配优先级。

Q

驱动器 (drive) 一种在可移动介质卷中存取数据的机械装置。

全局指令 (global directive) 应用于所有文件系统的存档程序和释放程序指令，位于第一个 `fs =` 行之前。

R

RAID 独立磁盘冗余阵列 (Redundant array of independent disks)。一种使用若干独立磁盘来可靠地存储文件的磁盘技术。它可以在单个磁盘出现故障时防止数据丢失；提供容错磁盘环境；以及提供比单个磁盘更高的吞吐量。

RPC 远程过程调用 (remote procedure call)。 NFS 用以实施用户网络数据服务器的底层数据交换机制。

软限制 (soft limit) 对于磁盘限额而言，是指用户可以临时超量使用的文件系统资源 （块或索引节点）的阈值限制。超过软限制时，系统会启动一个计时器。当超过软限制的时间大于指定时间 （默认值为一个星期）时，用户将不能再超量使用系统资源，直到减少文件系统的使用至低于软限制的水平。

S

samfsdump 一个程序，用于为给定的文件组创建控制结构转储文件并复制所有控制结构信息。它与 UNIX `tar`(1) 实用程序类似，但它通常不复制文件数据。

samfsrestore 一个程序，用于从控制结构转储文件中恢复索引节点和目录信息。

SCSI
小型计算机系统接口 (Small Computer System Interface)。一种电子通信技术规格，通常用于磁盘驱动器、磁带驱动器和自动化库等外围设备。

Sun SAM-QFS
Sun SAM-QFS 软件将 Sun StorEdge SAM-FS 软件和 Sun StorEdge QFS 文件系统组合在一起。 Sun SAM-QFS 不仅为用户和管理员提供了高速的标准 UNIX 文件系统接口，而且还提供了存储及存档管理实用程序。它可以使用 Sun StorEdge SAM-FS 命令集中的许多命令和标准 UNIX 文件系统命令。

Sun SAM-Remote 服务器 (Sun SAM-Remote server)
Sun SAM-Remote 服务器既是一台性能完备的 Sun StorEdge SAM-FS 存储管理服务器；又是一个 Sun SAM-Remote 服务器后台程序，用于定义 Sun SAM-Remote 客户端之间所共享的库。

Sun SAM-Remote 客户端 (Sun SAM-Remote client)
Sun SAM-Remote 客户端是一种 Sun StorEdge SAM-FS 系统，用于存放包含多种伪设备的 Sun SAM-Remote 客户端后台程序。它可能有 （也可能没有）自己的库设备。客户端用来存储一个或多个存档副本的存档介质由 Sun SAM-Remote 服务器来决定。

Sun StorEdge QFS
一种高速 UNIX 文件系统，它通过将文件系统元数据和文件数据存储在不同的设备上，来分离这两种数据。 Sun StorEdge QFS 软件可以控制对所有已存储的文件、以及所有配置在主配置文件 (`mcf`) 中的设备的访问。

Sun StorEdge SAM-FS
Sun 存储和存档管理器文件系统 (Sun Storage and Archive Manager File System)。 Sun StorEdge SAM-FS 软件可以控制对所有已存储的文件、以及所有配置在主配置文件 (`mcf`) 中的设备的访问。

设备日志 (device logging)
一项可配置的功能，用于提供设备专用的错误信息，以供分析设备问题。

设备扫描程序 (device scanner)
Sun StorEdge SAM-FS 文件系统中的一种软件，用于定期监视所有手动安装的可移动设备，并检测是否存在可供用户或其它进程请求的已安装卡盒。

设备系列集 (family device set)
参阅 “系列集”。

释放程序 (releaser)
一个 Sun StorEdge SAM-FS 组件，用于标识存档文件和释放其磁盘高速缓存副本，从而增大可用的磁盘高速缓存空间。释放程序可以自动将在线磁盘存储量调整到阈值上限和阈值下限。

释放优先级 (release priority)
一种计算文件系统中文件的释放优先级的方法。具体计算方法是：用各种加权数乘以相应的文件属性，然后对这些结果求和，求和结果即为文件的释放优先级。

数据设备 (data device)
: 对于 Sun StorEdge QFS 或 Sun StorEdge SAM-FS 文件系统，是指存储文件数据的一台或一组设备。

索引节点 (inode)
: inode 是 Index node 的缩写。是文件系统用来描述文件的一种数据结构。索引节点描述了与文件关联的所有属性（名称属性除外）。属性包括所有权、存取、权限、大小和文件在磁盘系统上的位置。

索引节点文件 (inode file)
: 文件系统中的一个特殊文件 (`.inodes`)，包含文件系统中存储的所有文件的索引节点结构。所有 Sun StorEdge QFS 和 Sun StorEdge SAM-FS 索引节点的大小都为 512 字节。索引节点文件属于元数据文件，在 Sun StorEdge QFS 文件系统中，元数据文件与文件数据分开存储。

T

tar
: 磁带存档 (Tape archive)。 Sun StorEdge SAM-FS 用于存档映像的一种标准文件和数据记录格式。

TCP/IP
: 传输控制协议 / 网际协议 (Transmission Control Protocol/Internet Protocol)。网际协议负责主机之间的寻址和路由以及数据信息包传递 (IP)，而传输控制协议负责在各个应用点之间可靠地传递数据 (TCP)。

V

VSN
: 卷序列名 (Volume serial name)。如果用户将数据存档至可移动介质卡盒，则 VSN 是指写入卷标中的磁带和光盘的逻辑标识。如果用户将数据存档至磁盘高速缓存，则它表示该磁盘存档组的唯一名称。

W

WORM
: 单次写入多次读取 (Write once read many)。一种介质存储类别，只能写入一次，但可以多次读取。

网络连接自动化库 (network-attached automated library)

由不同制造商生产的库，如 StorageTek、 ADIC/Grau、 IBM 或 Sony 等，它们由制造商提供的软件包控制。Sun StorEdge SAM-FS 文件系统通过自动化库的专用 Sun StorEdge SAM-FS 介质更换器后台程序，与供应商软件进行接口。

伪设备 (pseudo device)

未关联任何硬件的软件子系统或驱动程序。

文件系统 (file system)

一种由文件和目录组成的多层结构集合。

文件系统专用指令 (file system specific directive)

位于全局指令后面的存档程序和释放程序指令，专用于特定的文件系统，以 `fs =` 开头。文件系统专用指令的应用范围到出现下一个 `fs =` 指令行或文件末尾结束。如果多个指令影响到一个文件系统，则文件系统专用指令会取代全局指令。

X

限额 (quota)

允许用户使用的系统资源量。

小型计算机系统接口 (Small Computer System Interface)

参阅 SCSI。

循环 (round robin)

一种按顺序将全体文件写入到多个逻辑磁盘的数据存取方法。当将单个文件写入磁盘时，这个文件的全部内容将写入至第一个逻辑磁盘。第二个文件将写入下一个逻辑磁盘，依此类推。每个文件的大小决定着 I/O 的大小。

默认情况下，除非有拆分组，否则 Sun StorEdge QFS 和 Sun StorEdge SAM-FS 文件系统将实施拆分式数据存取。如果指定循环存取方法，则采用循环方法存取文件。如果文件系统包含不匹配的拆分组，系统将不支持拆分功能，并且强制实行循环存取方法。

另请参阅 “磁盘拆分”和 “拆分”条目。

Y

以太网 (Ethernet) 一种局域分组交换网络技术。它的最初设计是使用同轴电缆，随着技术的进步，价格便宜的屏蔽双绞线目前已逐渐取代了同轴电缆。以太网是指运行速度为 10 MB/s 或 100 MB/s 的局域网。

硬限制 (hard limit) 对于磁盘限额而言，是指用户不能超量使用的文件系统资源 （块或索引节点）的最大限制。

预分配 (preallocation) 在磁盘高速缓存中预先保留一定数量的连续空间以备写入文件的过程。这可以确保获得连续的空间。只能对大小为零的文件执行预分配操作。也就是说，您只能对大小为零的文件运行 `setfa -l` 命令。有关详细信息，请参阅 `setfa`(1) 手册页。

元数据 (metadata) 与数据有关的数据。元数据是指用于在磁盘上查找某个文件的具体数据位置的索引信息。它由以下各项有关信息组成：文件、目录、访问控制列表、符号链接、可移动介质、分段文件和分段文件索引。元数据用于确定数据的位置。在丢失数据时，您必须先恢复元数据才能恢复丢失的数据，因此，元数据必须得到保护。

元数据设备 (metadata device) 用于存储 Sun StorEdge QFS 文件系统元数据的独立设备，如固态磁盘或镜像设备等。分开存放文件数据和元数据可以提高系统的性能。在 `mcf` 文件中，元数据设备被声明为 `ma` 文件系统中的 `mm` 设备。

远程过程调用 (remote procedure call) 参阅 RPC。

阈值 (threshold) 一种为在线存储设备定义适当的可用存储空间的机制。阈值用于设置释放程序的存储目标。另请参阅 “磁盘空间阈值”。

Z

在线存储设备 (online storage) 可以即时访问的存储设备，如磁盘高速缓存等。

直接 I/O (direct I/O) 一种用于大型块对齐连续 I/O 的属性。`setfa`(1) 命令的 `-D` 选项即为直接 I/O 选项。它用于为文件或目录设置直接 I/O 属性。如果应用到目录，则直接 I/O 属性可以继承。

直接访问 (direct access)

一种文件属性 （永远不必登台），表示可从存档介质直接访问近线 (nearline) 文件，而无需检索磁盘高速缓存。

直接连接库 (direct-attached library)

是指通过 SCSI 接口直接连接到服务器的自动化库。一种 SCSI 连接库， Sun StorEdge SAM-FS 软件使用 SCSI 自动化库的标准对它进行直接控制。

自动化库 (automated library)

一种自动控制的设备，它可在无操作人员参与的情况下，载入和卸载可移动介质卡盒。自动化库包括两个部分：一个或多个驱动器；以及用于将卡盒移入或移出存储端口和驱动器的传输装置。

租借 (lease)

在 Sun StorEdge QFS 共享文件系统中，租借用于向客户端主机授予权限，使其可在租借有效期内对文件进行操作。元数据服务器向每一台客户端主机发放租借。它随时可以根据需要更新租借，以使客户端主机能够继续对文件进行操作。

索引

D

E

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z