IBM® 分布式计算环境版本 3.2 AIX® 和 Solaris 版

管理指南 -- 核心组件

IBM® 分布式计算环境版本 3.2 AIX® 和 Solaris 版

管理指南 -- 核心组件

注

在使用此文档之前，请阅读第471页的『附录E. 声明』中的一般信息

第一版（2001 年 7 月）

本版本适用于 *IBM 分布式计算环境 AIX 和 Solaris 版*版本 3.2 及其所有后续发行版和修订版，直到在新版本和技术通讯中另有声明为止。

可通过您所在地区的 IBM 代理或 IBM 分支部门订购出版物。以下地址不提供出版物。

IBM 欢迎您的建议。请将您的建议寄到以下地址:

International Business Machines Corporation
Department VLXA
11400 Burnet Road
Austin, Texas
78758

当您向 IBM 发送信息时，就授予了 IBM 非独有的权利，IBM 有权以它认为适当的任何方式使用或散发该信息，而不必对您承担任何责任。

本文档及其相关软件部分来自于以下机构提供的资料:

目录

图

表

关于本书

IBM DCE 版本 3.2 AIX 和 Solaris 版: 管理指南--核心组件 提供了使您能够管理 IBM® 分布式计算环境 (DCE) 的概念和步骤。本指南介绍了基本 DCE 术语。在 *IBM DCE 版本 3.2 AIX 和 Solaris 版: DCE 简介*中提供了所有 DCE 文档的词汇表。*IBM DCE 版本 3.2 AIX 和 Solaris 版: DCE 简介* 将有助于您对 DCE 技术的更深层次理解，它还描述了支持 DCE 的文档集。

读者

本指南是为那些具有 AIX® 或 Solaris 环境管理经验的系统和网络管理员所著。

适用性

本版本适用于现在提供的 IBM DCE 3.2 以及相关更新版本。请参阅软件许可证以获取详细信息。

用途

本指南的用途是帮助系统和网络管理员规划、配置并管理 DCE。阅读本指南后，您将理解系统管理员规划 DCE 所需要完成的任务。一旦在系统上构建了 DCE 源代码，请使用本指南辅助您安装可执行文件并配置 DCE。请参阅 *IBM DCE Version 3.2 for AIX: Quick Beginnings* 或 *IBM DCE 版本 3.2 Solaris 版: 快速入门*以获取在您的平台上安装并构建 DCE 源代码的信息。

文档用法

*IBM DCE 版本 3.2 AIX 和 Solaris 版: 管理指南*包含两本书，每本书分为如下部分:

- *IBM DCE 版本 3.2 AIX 和 Solaris 版: 管理指南--简介*
  - 第一部分 DCE 管理介绍
  - 第二部分 配置与设置 DCE
- *IBM DCE 版本 3.2 AIX 和 Solaris 版: 管理指南--核心组件*
  - 第1页的『第1部分 DCE 控制程序』
  - 第55页的『第2部分 DCE 管理任务』
  - 第81页的『第3部分 DCE 主机和应用程序管理』
  - 第121页的『第4部分 单元目录服务』
  - 第205页的『第5部分 DCE 分布式时间服务』
  - 第249页的『第6部分 DCE 安全性服务』

相关文档

关于分布式计算环境的附加信息请参阅下列文档:

- *IBM DCE 版本 3.2 AIX 和 Solaris 版: DCE 简介*
- *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference*
- *IBM DCE Version 3.2 for AIX and Solaris: Application Development Reference*
- *IBM DCE Version 3.2 for AIX and Solaris: Application Development Guide--Introduction and Style Guide*
- *IBM DCE Version 3.2 for AIX and Solaris: Application Development Guide--Core Components*
- *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*
- *IBM DCE Version 3.2 for AIX and Solaris: Application Development Guide--Directory Services*
- *OSF DCE/File-Access Administration Guide and Reference*
- *OSF DCE/File-Access User's Guide*
- *IBM DCE Version 3.2 for AIX and Solaris: Problem Determination Guide*
- *OSF DCE Testing Guide*
- *OSF DCE/File-Access FVT User's Guide*
- *Application Environment Specification/Distributed Computing*
- *IBM DCE 版本 3.2 AIX 版: 发行说明*
- *IBM DCE 版本 3.2 Solaris 版: 发行说明*

有关 DCE 3.2 AIX 和 Solaris 版文档的详细描述，请参阅 *IBM DCE 版本 3.2 AIX 和 Solaris 版: DCE 简介*。

印刷和按键约定

本指南使用如下印刷约定:

黑体 **黑体**字或字符表示确切使用的系统元素，如命令、选项和路径名。

斜体 *斜体*字或字符表示必须提供的变量值。*斜体*类型还用来引入新的 DCE 术语。

等宽 系统在等宽字体中显示的示例和信息。

[ ] 方括号中包含格式和语法说明的可选项。

{ } 花括号中包含格式和语法说明的必选项列表。

| 竖线栏分隔选项列表中的选项。

< > 尖括号中包括键盘上的按键名称。

... 省略号表示可一次或多次重复前述项。

本指南使用如下按键约定:

<Ctrl-*x*> 或 ^*x*

后跟键名标记 <Ctrl-*x*> 或 ^*x* 表示 control 字符序列。例如，<Ctrl-C> 表示按下 control 时再按 <C>。

<Return>

标记 <Return> 是指终端或工作站上标注为 Return、Enter 或左箭头的键。

问题报告

如果有任何关于软件或文档的问题，请与软件供应商的客户服务部联系。

DCE 文档中目录和文件的路径名

要获取本指南中涉及的目录和文件的路径名列表，请参阅 *IBM DCE 版本 3.2 AIX 和 Solaris 版: 管理指南--简介*和 *OSF DCE Testing Guide*。

第1部分 DCE 控制程序

第1章 DCE 控制程序介绍

DCE 是一组集成的服务，用于支持异构联网计算机之间分布式应用程序的开发和执行。每个 DCE 环境（也称为一个*单元*）至少维护下列 DCE 核心服务:

- DCE 线程
- DCE 主机服务
- DCE 单元目录服务
- DCE 时间服务
- DCE 安全性服务

除 DCE 线程之外，所有核心服务都需要某种方式的管理。有些服务（例如 CDS 和 DCE 安全性服务）相对于那些一旦设置好就实际上不再需要介入的服务（比如 DCE 时间服务）来说需要更多的管理。如果您的 DCE 仅由少数计算机及其用户组成，可以通过登录到不同主机并执行任何必需任务的方式来管理用户、程序和主机系统对命名、时间和安全性的需求。但是大多数的单元将由很多，可能是成百上千的，计算机及其用户组成。

所以在这样的大型单元中，核心服务将很可能非常广泛而且复杂，有些服务将被复制或划分到多个异构的系统之中。某些服务，例如 DCE 主机服务，将存在于单元中的每个计算机上。如此大规模的操作需要一个管理界面，无论管理功能位于何处，从单元的任何一个端点或每个端点都能够对 DCE 的管理功能进行一致、相同的访问。这意味着无论从任何平台上执行，管理操作必须是一致的并且是可预测的。

DCE 控制程序 (**dcecp**) 满足了这一要求，提供从一个 DCE 单元的任意端点对几乎所有 DCE 管理功能的一致、可移植、可扩展和安全的访问。**dcecp** 实现了所有原来用各种组件控制程序所完成的操作。

通过提供一组*任务对象*来执行复杂 DCE 操作，**dcecp**将管理进一步简化。例如，将一个主机添加到单元中时需要在注册表中添加一个主机主体，将主体添加到各种安全性组和组织，创建帐户，将主机信息放置到 CDS ，并可能在 CDS 目录中设置一些 ACL。所有这些操作可以使用单一任务对象来完成。

注: **dcecp** 被设计成支持多种语言环境。它替换了一些老的控制程序 (cdscp, dtscp, rpccp, acl_edit, rgy_edit, sec_admin)。这些老的程序没有被设计成国际通用，在非英语环境中使用时会出现一些异常或不期望的结果。

尽管 **dcecp** 支持非英语数据，不过也有一些限制。**dcecp** 字符串处理命令，例如字符串范围命令，有基于字节而不是基于字符的语义。在处理 DCE 可移植字符集之外的字符时可能产生不期望的结果。

灵活、可移植和可扩展的管理

dcecp 建立在一个称为 Tcl（发音为 『tickle』）的可移植命令语言之上，Tcl 表示由美国加利福尼亚州大学伯克利分校的 John K. Ousterhout 所开发的工具命令语言（Tool Command Language）。大多数计算机提供某种类型的命令语言，以便用户以一种灵活而可扩展的方式来访问并使用系统的能力。例如，许多 UNIX® 系统提供外壳语言解释器，

DEC 公司的 OpenVMS 操作系统提供 Digital Command Language(DCL)。但是这些命令语言并不总是可移植的。基于一种命令语言的命令和脚本在另外一种语言环境中可能无法工作。

另一方面，Tcl 是一种在每个安装了 DCE 的系统上运行的平台无关命令语言。Tcl 命令解释器和使用它的 DCE 控制程序是作为 DCE 软件的一部分提供的。

DCE 控制程序和 DCE 控制程序语言的可用性 DCE 管理员提供了非常重要的好处:

- 可以从单一管理界面执行所有日常的 DCE 操作。
- 从任何 DCE 平台都可以一致而统一地执行大多数 DCE 管理操作，使得 DCE 管理员可以在单元的任何 DCE 系统中管理几乎所有 DCE 操作。非 UNIX 系统的 DCE 平台可能无法处理全部的 DCE 控制程序文件操作。
- **dcecp** 提供类似 **clearinghouse**、**principal** 和 **endpoint** 名称的管理*对象*。这一直接处理使得 DCE 管理更加直观而且一致。虽然目前只是看上去是面向对象的，但是这是迈向真正面向对象式管理界面的重要一步。
- *任务对象*（执行复杂 DCE 操作的高级 **dcecp** 脚本）减少了对 DCE 管理员的培训要求。员工不必在成为 DCE 专家后才能执行日常的 DCE 管理任务。
- 通过使用 **dcecp** 操作和 Tcl 提供的更多的命令，可以把提供的任务对象改作新的用途，或者编写新任务对象或脚本。
- **dcecp** 语言允许使用变量、**if** 语句、循环函数和其它增强操作能力的编程操作。例如，循环函数允许对多个对象，例如用户、服务器或 CDS 项，进行重复操作。
- 因为脚本可以不必修改地移到外部平台上，所以管理员可以方便地共享他们的工具。例如，有多个单元的企业可以使用 **dcecp** 脚本将一个公共单元配置传播到整个企业。

DCE 控制程序是一个管理的接口，用于对 DCE 核心组件的大多数特性进行管理。不过无法用 **dcecp** 管理 DCE 的各个方面。例如，**dcecp** 不能控制 DFS™。

第1页的『第1部分 DCE 控制程序』 中的章节讨论如何使用 **dcecp** 管理 DCE 环境中的核心服务，还讨论如何通过在命令行中使用 Tcl 构造并将定制的操作编写成脚本使得操作完成更多的工作。我们并不提供对 Tcl 和用于 X11 窗口系统的配套工具箱（称为 *Tk*）的完整讨论。

DCE 管理对象

一个 DCE 单元由许多需要管理的部分组成，例如 CDS 服务器 (*clearinghouse*)、DTS 时钟和服务器位置信息都是 DCE 单元中需要进行某种管理的实体。DCE 控制程序将所有 DCE 管理实体作为单独的管理对象进行处理。

通过调用实体的*对象*名加上某些操作来操作一个实体。例如，要检查一个 DTS 时钟的时间，调用对象的名称 (**clock**) 和期望的操作 (**show**)，如下所示:

```
dcecp> clock show
1994-09-23-10:46:42.016-04:00I-----
dcecp>
```

DCE 中的每个管理实体在 DCE 控制程序中都有一个对应的管理对象。例如，可以使用 **clearinghouse** 对象来管理 CDS 中央交换库操作，使用 **server** 对象来管理应用程序服务器及其在 DCE 主机上的配置信息，使用 **utc** 对象比较并操纵时间信息，使用 **user**

任务对象管理一个 DCE 单元中的用户。这些示例只代表几个 **dcecp** 管理对象。*IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference* 中列出了所有对象。

使用 DCE 控制程序

本节简要说明如何启动和停止 DCE 控制程序以及如何执行操作。这些主题的附加信息都包含在 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference* 中。

启动和停止 dcecp

可以在操作系统提示符下或 DCE 控制程序中直接输入 **dcecp** 操作。如果只执行一或两个简单 **dcecp** 操作，可以直接在操作系统提示符下进行调用。

如果要进行多个操作，可以调用 DCE 控制程序，然后在 **dcecp** 提示符下输入操作。这种方式有几种优点。

- 对于多个操作而言更有效，因为相对于每个单独的操作来说 **dcecp** 只初始化了一次。
- 控制程序将操作存储在历史设施中以便重新调用并重用。
- 不需要在每个操作前添加 **dcecp** 命令，从而避免额外的击键。

下列示例显示如何调用 DCE 控制程序并执行一个 **directory** 操作:

```
% dcecp
dcecp> directory create /.:/hosts/appserver2
dcecp>
```

在用完 DCE 控制程序后，使用 **exit** 或 **quit** 操作来停止程序并返回到操作系统提示符。下列示例说明 **exit** 操作的用法:

```
dcecp> exit
%
```

调用 dcecp 操作

如果要执行单一 **dcecp** 操作，可以在操作系统提示符下直接调用，只要在期望的操作前加上 **dcecp** 命令以及 **-c**（命令行操作）标志即可，如下:

```
% dcecp -c directory list /.:/subsys -simplename
HP applications dce sales eng admin accts
% dcecp -c cell show
{secservers
 /.../my_cell.goodco.com/subsys/dce/sec/master}
{cdsservers
 /.../my_cell.goodco.com/hosts/krypton}
{dtsservers
 /.../my_cell.goodco.com/hosts/mars}
{hosts
 /.../my_cell.goodco.com/hosts/earth
 .../my_cell.goodco.com/hosts/jupiter
 /.../my_cell.goodco.com/hosts/kyrpton
 /.../my_cell.goodco.com/hosts/mars
 /.../my_cell.goodco.com/hosts/mercury
 /.../my_cell.goodco.com/hosts/neptune
 /.../my_cell.goodco.com/hosts/pluto
 /.../my_cell.goodco.com/hosts/saturn
 /.../my_cell.goodco.com/hosts/uranus
 /.../my_cell.goodco.com/hosts/venus}
%
```

也可以输入一些受限制的、由分号分隔并由 ""（双引号）括起的多个操作。下列示例向注册表添加一个主体，然后检查添加的主体:

```
% dcecp -c "principal
create S_Preska ; principal show S_Preska"
{fullname {}}
{uid 28}
{uuid 0000001c-dc77-21cd-b700-0000c08adf56}
{alias no}
{quota unlimited}
%
```

在使用带有 **-c** 选项的 **dcecp** 命令输入多个操作时必须非常小心，因为操作结果返回到 **dcecp** 解释器，而不是外壳解释器。类似下列的操作仅仅将最后一个操作 (group list users) 的结果返回到外壳:

```
% dcecp -c "group list staff; group list managers; group list users"
/.../ward_cell.osf.org/P_Pestana
/.../ward_cell.osf.org/R_Parsons
/.../ward_cell.osf.org/L_Jones
/.../ward_cell.osf.org/S_Preska
/.../ward_cell.osf.org/N_Long
/.../ward_cell.osf.org/D_Witt
/.../ward_cell.osf.org/C_Pilat
 .
 .
 .
%
```

可以这样来克服这种特别的问题:

```
% dcecp -c "puts [group list staff]; puts [group list manager];
puts [group list users]"    .    .    . %
```

要调用一个 **dcecp** 脚本，必须省略 **-c** 选项，但是要包含脚本的名称。下列示例调用一个以字母次序显示单元中所有主机名称的脚本:

```
% dcecp list_hosts
earth
jupiter
krypton
mars
mercury
neptune
planets
pluto
saturn
uranus
venus
%
```

在需要调用复杂的或多个操作时，可能想要从 **dcecp** 中调用。这一程序提供了一个方便的历史设施并有命令行编辑能力，非常便于重新调用和重用前面的操作。下列示例中的操作调用 **dcecp** 并向 DCE 单元添加一个新用户:

```
% dcecp
dcecp> principal create J_Jones
dcecp> group add users -member J_Jones
dcecp> organization add staff -member J_Jones
dcecp> account create J_Jones -group users -organization staff \
> -password change.me -mypwd mxyzptlk
dcecp>
```

在交互式使用时，所有 **dcecp** 对象、操作和选项名称都可以缩写成最短的唯一字符串。在选择名称时采用了这种最短原则，唯一的缩写通常都不超过一个或两个字符。

应当避免在脚本中使用对象或命令缩写，因为这样会限制脚本的可移植性。用户定义的命令可能会影响缩写的唯一性，导致歧义的命令名或对象名。

进一步使用 dcecp

DCE 控制程序接受从简单到复杂的各种命令，更复杂的命令提供更强的能力和适用性。虽然简单的命令最容易组成，但是它们也同样受到限制，通常被限制成对单一对象执行一个操作。所以，虽然总是可以通过输入简单命令来完成，有时会发现需要对几个甚至许多对象重复执行操作，或者只需要在特定条件下执行某些操作。比如，可能需要向 CDS 中添加某些项，而这些项只有在其它特定的项已经存在于 CDS 中时才需要添加。利用 Tcl 中模仿多种编程语言和外壳语言中常见部分的内建命令，**dcecp** 可以满足这一要求。

DCE 控制程序包含许多用于控制命令的执行的类 C 语言的构造。如用于条件执行的 **if** 语句，用于在不同条件下重复操作的循环命令，例如 **while**, **for** 和 **foreach** 等，用于针对不同模式测试值的 **case** 命令，以及用于编写定制命令的 **proc** 等等。

DCE 控制程序也包含其它句法元素，例如 ″″（引号）、{ }（花括号）、[ ]（方括号）和 \（反斜杠），用来对元素进行组合，或控制对特殊字符的解释。

虽然许多特性都是为了在脚本中使用而设计的，但是在交互式操作的过程中，您可能发现自己也在使用某些构造和元素（特别是引号、花括号、方括号和反斜杠）。有时必须决定哪一种方式更有意义，即交互式地执行操作还是使用脚本来完成。通常，操作的复杂程度和重用的可能性可以帮助您做出决定。

现在看一些说明 DCE 控制程序和 Tcl 基础概念的简单示例。某些 **dcecp** 操作可以是非常直接的，就像:

```
dcecp> account modify N_Long -expdate
1996-06-30
dcecp>
```

这一操作更改 DCE 安全性服务注册表中的信息。这里改变了命令行中所指定主体 (**N_Long**) 的帐户失效日期。虽然对一两个主体执行这一操作很简单，但是如果要修改更多主体的帐户就困难得多。

设想您的组织雇佣了六名临时员工，而且他们参与的项目将持续三个月。您可以用一个 **dcecp foreach** 命令来循环操作列表中的每一项，而不是将 **account modify** 操作执行六次:

```
dcecp> foreach i {N_Long L_Jones P_Sawyer \
> D_Witt M_Dougherty S_Preska} { \
> account modify $i -expdate 1996-06-30 }
dcecp>
```

在示例中，**foreach** 循环命令有三个参数: 变量，列表和循环体。变量 **i** 按顺序替代列表（**N_Long**, **L_Jones** 等等）中的每个项， **foreach** 命令对列表中的每个项执行循环体 (**account modify $i -expdate 1996-06-30**)，循环体中的变量 **$i** 按顺序取列表中每个主体名称的值，直到取完所有值为止。如需有关循环命令的详细信息，请参阅第31页的『使用循环控制脚本执行』。

这个示例同时表明了其它几个重要语法规则。DCE 控制程序使用 { }（花括号）来确定命令参数的开始和结束位置，例如脚本体。例如，**foreach** 命令有三个参数：变量、列表和脚本命令体。通常命令参数是以空格分隔的。为了避免 **dcecp** 将列表元素间的空格解释为分隔符，使用花括号将列表括起，以禁止对空格的特殊解释。这样所有的列表元素表现为一个参数。同理，使用花括号来括起脚本体中各个单独的元素。

花括号可以帮助 **dcecp** 确定一个命令是否已经完成，一个未完成的命令将带有更多的开始花括号而不是结束花括号。在第一行结束时缺少一个结束花括号通知 **dcecp** 还有更多的命令正在输入，这样 **dcecp** 显示第二个提示符 (>)。同样，第二行行尾的开始花括号表示仍然没有完成命令输入。这允许直接换多行而不必使用一个 \ (反斜杠）行换字符。在第三行结尾的结束花括号后面按 **<Return>** 之后，DCE 控制程序执行这一命令。第19页的『第2章 使用 DCE 控制程序命令语言』包含有关花括号的更多详细信息。

现在假设组织有五十个临时员工（都在一个称为 **temps** 的组中）而不是六个，每个都需要添加三个月帐户延长。这里将同样使用 **foreach** 命令，不过要用 **dcecp group list temps** 操作来产生一个列表，而不是将五十个主体直接写在列表中:

```
dcecp> foreach i [group list temps] {
> account modify $i -expdate 1996-06-30 }
dcecp>
```

在示例中，我们将 group list temps 操作放到 [ ]（方括号）中，这叫做*命令替换*，该技术用命令的返回结果替换方括号中的命令。**group list temps** 操作的结果产生一个有效的 Tcl 列表，如下:

```
dcecp> group list temps
N_Long
L_Jones
P_Sawyer
D_Witt
M_Dougherty
S_Preska
 .
 .
 .
J_Jones
```

这里，我们对 **dcecp** 的一些实际使用进行了简单浏览，当然还有许多尚未涉及。在下一章中，我们将详细介绍可能用作 DCE 管理的一些 **dcecp** 操作。请注意，**dcecp** 是基于 Tcl 的，而 Tcl 带有其它我们不予讨论的命令和命令的变化。因此请确认您可以得到 Tcl 的标准出版物，以获取所有命令的详细信息。

何时应当使用交互式命令或脚本

在应当交互式地输入命令还是应当使用脚本之间并没有绝对的划分线。通常情况下，简单的操作，即那些执行一两个任务的操作最适于交互式使用。下列示例是典型的交互式操作:

```
dcecp> directory create /.:/printers
dcecp> account show w_shakespeare
dcecp> server start /.:/hosts/curley/config/srvrconf/BBSserver
```

下一个示例有些复杂，所以首先应当选择以脚本运行:

```
foreach i [group list temps] {
   account modify $i -expdate 1996-06-30}
```

将频繁使用的操作保存为一个脚本（在一个文件）有多个优点，这样有助于将重复或复杂的任务自动化，并且可以保留下来，以备将来进行修改并用在其他的情况下。无论选择哪个方式，在您习惯了使用 **dcecp** 和 Tcl 之后，可能发现自己可以直接交互地输入复杂的操作。如需有关如何创建和调用脚本的详细信息，请参阅第15页的『定制 dcecp 会话』。

编辑命令行

我们已经看到了一些输入交互式 **dcecp** 命令的基本方法。现在我们需要编辑正在输入的命令或重用并修改以前输入的命令。DCE 控制程序提供多种编辑命令的方法。可以使用命令行编辑功能来编辑当前的命令行。可以使用 **history** 命令来重用、编辑并重发一个以前用过的命令。

编辑当前命令行

在将命令行发送到 **dcecp** 之前，可以通过输入类似 `ksh` 或 `emacs` 编辑命令的控制字符或换码序列的方法来编辑命令行。一个*控制字符*显示为 **<Ctrl-*x*>**，这里 *x* 是一个字母，控制字符是在按下 **<Ctrl>**（或 **<Control>** ）的同时按字母键输入的。例如，**<Ctrl-A>** 是同时按 **<Ctrl>** 和 **<A>**。按 **Escape** 以及一个或多个字符可以输入一个*换码序列*。在换码序列中，**<Escape>** 被引用为 **ESC**，例如 **<ESC f>**。大小写在换码序列中有效（不像控制字符对大小写不加区分），**<ESC F>** 和 **<ESC f>** 是不同的。

不仅在行首，在行的任意位置都可以输入编辑命令。另外，回车可以在行的任意位置输入，而不必一定在行尾。

大多数编辑命令接受重复计数 *n*，*n* 是一个数字。这样输入一个重复计数：按 **<Escape>**，输入数字，然后输入要执行的命令。例如 **<ESC 4><Ctrl-f>** 正向移动四个字符。下列带有 [*n*] 标记的描述表示该命令接受重复计数。

接受的控制字符如下:

<Ctrl-A>
移动到行首

<Ctrl-B>
向左（反向）移动 [*n*]

<Ctrl-D>
删除字符 [*n*]

<Ctrl-E>
移动到行尾

<Ctrl-F>
向右（正向）移动 [*n*]

<Ctrl-G>
响铃

<Ctrl-H>
删除光标前的字符（**<Backspace>**）[*n*]

<Ctrl-I>
使文件名完整（**<Tab>**），见列文

<Ctrl-J>
完成本行 (**<Return>**)

<Ctrl-K>
删到行（或列 [*n*] ）尾

<Ctrl-L>
重新显示行

<Ctrl-M>
完成本行 (alternate **<Return>**)

<Ctrl-N>
获取历史中的下一行 [*n*]

<Ctrl-P>
获取历史中的前一行 [*n*]

<Ctrl-R>
反向（如果 [*n*] 则正向）搜索历史以查找文本，如果文本以向上键开始，必须是起始行

<Ctrl-T>
移动字符

<Ctrl-V>
插入下一个字符，即使这是一个编辑命令

<Ctrl-W>
擦除到标记

<Ctrl-X><Ctrl-X>
交换当前位置和标记

<Ctrl-Y>
粘贴删除的文本

<Ctrl-[>
开始一个换码序列 (**<Escape>**)

<Ctrl-]>*c*
正向移动到下一个字符 *c*

<Ctrl-?>
删除光标前的字符 (**<Delete>**) [*n*]

接受的换码序列如下:

<ESC><Ctrl-H>
删除前一个单词 (**<Backspace>**) [*n*]

<ESC DEL>
删除前一个单词 (**<Delete>**) [*n*]

<ESC SPC>
设置标记 (**<Spacebar>**)，见 **<Ctrl-X><Ctrl-X>** 和 **<Ctrl-Y>**

<ESC .>
从上一行获取最后一个（或第 [*n*] 个）单词

<ESC ?>
显示可能的完成，见下文

<ESC <>
移动到历史记录的开始

<ESC >>
移动到历史记录的结束

<ESC b>
反向移动一个单词 [*n*]

<ESC d>
删除光标后面的单词 [*n*]

<ESC f>
正向移动一个单词 [*n*]

<ESC l>
将单词转换为小写 [*n*]

<ESC u>
将单词转换为大写 [*n*]

<ESC y>
粘贴删除的文本

<ESC w>
使至标记之间的区域成为可粘贴的

<ESC *nn***>**
将重复计数设置为数字 *nn*

在某些情况下，现有终端键绑定的优先权高于 **dcecp** 控制键。特别是环境中用于 **erase**、 **kill**、**eof**、**intr**、**quit** 和 **susp** 的绑定永远优先于 **dcecp** 对相同控制键的绑定。在大多数情况下，控制键不会被终端解释，而是传递给 **dcecp**。一个例外是 **lnext**（下一个文字），表示引用下一个输入的字符。当输入绑定为 **lnext** 的控制键时，它将被终端解释，并将输入的下一个字符传递给 **dcecp**。

DCE 控制程序也提供文件名完成。假设根目录包含下列文件:

```
bin vmunix
core vmunix.old
```

如果输入 **rm /v** 然后按 **<Tab>**，命令处理器将尽可能对文件名称进行完成，自动添加 **munix**。因为示例名称不是唯一的，它将响铃。如果按 **<Escape>** 然后按 **?**（问号），将显示出两个选项。当输入小数点（使名称唯一）接着输入 **<Tab>** 之后，命令处理器完成这一文件名，如下列所示:

```
rm /v <Tab>munix.<Tab>old
```

示例中等宽字体表示由命令处理器自动输入的文本。

使用 history 命令编辑命令行

在输入交互式命令时，有时需要重用并重用一个原先输入的命令。假设列出一个 CDS 目录中的对象，然后修改了一个对象。现在需要再次列出那些对象以验证修改是否已经

生效。可以使用 **history** 命令来检索、编辑并重发一个以前使用过的命令。历史设施仅保存交互式命令，不保存脚本中发出的命令，也无法对其重用。

history 命令根据用途使用多种参数。不带参数输入 **history** 显示当前 **dcecp** 调用中输入的所有命令（称为*事件*），如下所示:

```
dcecp> history
     1  principal create wardr -fullname {Ward Rosenberry} \
          -quota unlimited
     2  group add users -member wardr
     3  organization add consultants -member wardr
     4  account create wardr -mypwd mxyptlk -password qwerty \
          -group users -organization consultants
     5  history
dcecp>
```

每个历史事件都独立于以前的事件。这表示，如果一个检索到的命令使用了一个变量，其当前值可能和它第一次输入时的值不同。**history** 命令本身也生成一个历史事件。

缺省情况下，历史列表保留最近的 20 条命令。可以使用 **history keep** 命令来增加或减少历史列表的长度。例如，下列命令增加历史列表的长度以保留最近的 50 条事件:

```
dcecp> history keep 50
dcecp>
```

可以用多种方式指定事件。正数指定相对于列表中最初事件的事件，负数指定相对于列表中最近事件的事件，也可以通过输入匹配以前事件部分或所有内容的字符来指定事件。

历史设施允许以许多方式重用以前的事件。下列讨论只涉及可以使用的一部分历史命令。

- 使用 **history redo** 命令可以不加修改地执行以前的命令:

  ```
  dcecp> history
              1 directory show /.:/printers
              2 object create /.:/printers/ascii_printer1
              3 object create /.:/printers/ascii_printer2
              4 object create /.:/printers/ascii_printer3
              5 history
  dcecp> history redo directory
  directory show /.:/printers
   .
   . [output omitted]
   .
  dcecp>
  ```

  在 history 命令中输入唯一的起始字符或单词，可以节省大部分输入。例如，可以这样输入上一个示例中的 **history redo directory** 命令:

  ```
  dcecp> hi r d
  directory show /.:/printers
   .
   . [省略输出]
   .
  dcecp>
  ```

  其它的重新执行命令包括 **!!**，表示重做最近的事件，**!***事件编号*以检索一个特定事件的。

- 可以使用 **history substitute** 命令来修订并执行以前的命令。该命令的常见用途是纠正输入错误。命令语法如下:

history substitute *old new* [*事件编号*]

如果省略*事件编号*，将重新执行最近的命令。检索到的命令中的 *old* 部分被替换为 *new* 信息:

```
dcecp> history
           1 directory show /.:/printers
           2 object create /.:/printers/ascii_printer1
           3 object create /.:/printers/ascii_printer2
           4 object create /.:/printers/ascii_printer3
           5 directory show /.:/printers
           6 history
dcecp> hi s printer3 printer4 -3
object create /.:/printers/ascii_printer4
dcecp>
```

也可以用 UNIX **csh** 外壳用户所熟悉的 ^*old*^*new* 语法检索并修订最近的命令，如下所示:

```
dcecp> ^4^5
object create /.:/printers/ascii_printer5
dcecp>
```

使用 dcecp 帮助设施

DCE 控制程序以多种方式提供帮助:

- 如果要查看 DCE 控制程序所提供对象的列表，在 **dcecp** 提示符下输入 help，如示例:

```
dcecp> help
The general format of all dcecp commands is as follows:
  dcecp> <object> <operation> [argument] [options]
In addition to all of the standard tcl commands, dcecp supports many
commands to administer DCE objects.  A dcecp object or task represents
a DCE entity.  Type 'man dcecp_<command>' for more information.
All
of the following dcecp objects and tasks require an operation:
  account        cdsalias       dts            log            rpcprofile
  acl            cdscache       endpoint       name           secval
  attrlist       cdsclient      group          object         server
  aud            cell           host           organization   user
  audevents      cellalias      hostdata       principal      utc
  audfilter      clearinghouse  hostvar        registry       uuid
  audtrail       clock          keytab         rpcentry       xattrschema
  cds            directory      link           rpcgroup
Miscellaneous commands perform specific functions.  Type 'man dcecp'
for more information.  These commands take no operation:
  echo     errtext  login    logout   quit     resolve  shell
To list all dcecp objects:                    dcecp> help -verbose
To list all operations an object supports:    dcecp> <object> help
To list all options for an object operation:  dcecp> <object> help <operation>
For verbose information on a dcecp object:    dcecp> <object> help -verbose
For the manual page of a dcecp object:        dcecp> man dcecp_<object>
dcecp>
```

- 如果要了解一个对象支持哪些操作，使用命令 *object* **operations**，将返回可以对该对象进行的操作。下列示例显示如何列出 **principal** 对象的可用操作:

```
dcecp> principal operations
catalog create delete modify show operations help
dcecp>
```

可以将该命令缩写为类似 **prin oper** 的形式以减少输入。

- 使用 *object* **help** 命令可以得到关于该对象及其操作更详细的帮助。下列示例返回对 **principal** 对象所支持操作的描述，每个操作一行:

```
dcecp> principal help
catalog             Returns all the names of principals in the registry.
create              Creates a DCE principal.
delete              Deletes a principal from the registry.
modify              Changes the information about a principal.
rename              Renames the specified principal.
show                Returns the attributes of a principal.
help                Prints a summary of command-line options.
operations          Returns a list of the valid operations for this command.
dcecp>
```

- 向 *object* **help** 命令添加一个 *operation* 参数，可以得到可用命令选项的信息。下列示例返回对 **principal create** 操作支持的选项的描述，每个选项一行:

```
dcecp> principal help create
-alias              Indicates the principal name is an alias of the uid.
-attribute          Specify principal attributes in an attribute list format.
-fullname           Fullname of the principal.
-quota              How many registry objects can the principal create.
-uid                User Identifier of the new principal.
-uuid               Orphaned UUID to be adopted by the principal.
dcecp>
```

- 使用一个 *object* **help -verbose** 命令可以获取关于对象本身的帮助。下列示例返回对 **principal** 对象的描述以及如何使用该对象的信息:

```
dcecp> principal help -verbose
This object allows manipulation of principal information stored
in the DCE registry.  The argument is a list of either relative or
fully-qualified principal names.  Specify fixed attributes using
attribute options or an attribute list.  Specify any extended attributes
using an attribute list.  Principal operations connect to a registry that
can service the request.  Specify a particular registry by setting the
_s(sec) convenience variable to be a cell-relative or global replica
name, or the binding of the host where the replica exists.  The
completed operation sets the _b(sec) convenience variable to the name
of the registry contacted.
dcecp>
```

- 最终，一些 POSIX 风格的系统将带有 **dcecp** 对象和 Tcl 摘要的参考页。每个 **dcecp** 对象有描述其本身的参考页以及可用操作的参考页。查看一个 **dcecp** 对象参考页的通用语法为:

  man *bject_name*

  下列示例显示如何调用 principal 对象的参考页。注意，在 **dcecp** 中可以使用 man。

```
dcecp> man principal
 .
 . [output omitted]
 .
dcecp>
```

  Tcl 参考页概述了 Tcl 内置的命令。可以在 UNIX 风格的系统上输入下列命令以查看 Tcl 的摘要参考页:

```
dcecp> man Tcl
 .
 . [output omitted]
 .
dcecp>
```

定制 dcecp 会话

DCE 控制程序包含一些用于执行大部分日常的 DCE 管理操作的命令、对象和任务脚本。然而，在使用 **dcecp** 接口的经验很丰富之后，您可能会发现需要添加一些新的命令和性能，或者对一些现有的进行定制。下列部分说明如何向 **dcecp** 会话添加新的脚本和对象。一个对象是脚本的正式实现，使用 **dcecp** 帮助系统，并采用 *object operation* 的格式。第19页的『第2章 使用 DCE 控制程序命令语言』和第45页的『第3章 编写脚本和 dcecp 对象』 说明了编写 **dcecp** 脚本和创建新对象的基本原理。

向 dcecp 会话添加脚本

一旦编写了脚本，您可以控制只允许某个人员使用，或者允许每个登录到主机的人员使用。这可以通过修改 **dcecp** 在初始化时调用的下列一个或多个文件来实现:

[info library]/init.tcl

这一文件首先被读取，包含用于该主机的标准 Tcl 初始化命令。影响主机中运行的所有 **dcecp** 实例。该文件包含 Tcl **unknown** 命令的定义和用于初始化所有 **dcecp** 对象的 **auto_load** 功能。管理员不应向这一文件添加 **dcecp** 的定制内容。

*dcelocal***/init.dcecp**

该文件包含主机中用于 **dcecp** 特定的启动信息。影响主机中运行的所有 **dcecp** 实例。实现操作和任务的 **dcecp** 脚本存储在 *dcelocal***/dcecp** 目录下。向这一文件中以过程格式添加的定制内容可供主机中所有 **dcecp** 用户使用。

$HOME/.dcecprc

这一可选文件存储影响个别 **dcecp** 用户（**.dcecprc** 文件的所有者）的定制。每个 DCE 用户可以维护一个 **.dcecprc** 文件，在其中存储操作的专用的过程或别名。在多管理员环境下，可以使用修改过的 **.dcecprc** 文件实现灵活的管理。例如，用于各个管理员的不同 **.dcecprc** 文件可以使用 **dcecp source** 命令来调用为特定管理内容定制的专用命令和任务脚本。

本节的剩余部分说明了一个简单的任务脚本，并且展示如何使脚本可供个人使用。这一示例以控制程序现有的、用于显示当前时间的 **clock** 对象开始。但是，时间只是本地主机时钟的 DTS 时间戳记:

```
dcecp> clock show
1994-10-03-10:22:59.991-04:00I-----
dcecp>
```

假设您创建了一个过程，该过程从一个 DTS 服务器获取时间戳记，并且和时间一起显示 DTS 服务器的名称，就像下列示例中调用的由用户创建的称为 **show_clock** 的过程:

```
dcecp> show_clock
Time on mars is        1994-09-30-15:03:43.979-04:00I-----
dcecp>
```

只要在该用户的 **.dcecprc** 文件中包含该过程，就可以使这一过程供其他用户使用。下列样本 **.dcecprc** 文件包含用户定制内容，包括 **_dcp_show_clocks** 过程和别名，这样用命令名 show_clocks 就可以调用这一过程。另一个叫做 **_dcp_whoami** 的过程显示当前登录身份的信息。请注意 **.dcecprc** 中操作的次序。 过程在文件的开始处定义。重命名和调用过程必须出现在定义了过程之后。

```
##
## Start up commands
##
# A simple command to rerun .dcecprc after modifications
proc .d {} {source $HOME/.dcecprc}
# Show your current login name and your current cell name.
proc _dcp_whoami {} {
  global _c _u
  return "You are '$_u' logged into '$_c'."
}
# Show the time on all of the dts servers running in your cell.
proc _dcp_show_clocks {} {
    set x [directory list /.:/hosts]
    foreach n $x {
        if {[catch {object show $n/dts-entity}] == 0} {
            set index [string last "/" $n]
            set y [string range $n [incr index] end]
            if {[catch {clock show $n/dts-entity} msg] == 0} {
                set i [expr 20 - [string length $y]]
                puts [format "Time on $y is %${i}s %s" " " \
                    [clock show $n/dts-entity]]
            } else {
                set i [expr 20 - [string length $y]]
                puts [format "Time on $y is %${i}s %s" " " \
                   "Server not responding."]
            }
        }
    }
}
# Give some procs usable names
rename _dcp_whoami whoami
rename _dcp_show_clocks show_clocks
# If I am authorized, say so
if {$_u != ""} {
  whoami
}
```

接近文件末尾的 **rename** 命令允许以更简单的命令名 **show_clocks** 和 **whoami** 调用过程 **_dcp_show_clocks** 和 **_dcp_whoami**。

如果登录到 DCE，在启动 **dcecp** 时，文件的最后部分调用 **_dcp_whoami** 过程。如果设置了 **_u** 便利变量，过程 **_dcp_whoami** 打印出当前登录的身份，如下所示:

```
% dcecp
You are 'principal_name' logged into 'cell_name'.
dcecp>
```

向 DCE 控制程序添加新的对象

如果已经将脚本作为正式 **dcecp** 对象的脚本编写，可以在存放任务对象的目录中包含这一新的对象，使其可用。在 UNIX 系统中，这一目录通常是 *dcelocal*/**dcecp**。作为规则，应当将新的对象添加到 DCE 单元的每个主机。第45页的『第3章 编写脚本和 dcecp 对象』 描述如何用 **dcecp hostdata** 对象将脚本或其它文件复制到单元中每个主机。

在安装新的脚本时，必须运行 **auto_mkindex** 实用程序以便主机中的其它用户也可以使用这一新对象。如需有关运行 **auto_mkindex** 实用程序的详细信息，请参阅 第45页的『第3章 编写脚本和 dcecp 对象』。

环境变量

环境变量是用户可以自行设置的 DCE 变量。如需有关 DCE 环境变量的详细信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Application Development Guide--Core Components*。

第2章 使用 DCE 控制程序命令语言

在第3页的『第1章 DCE 控制程序介绍』中简单介绍了使用 DCE 控制程序管理 DCE 环境的一些方法。本章讨论在编写 **dcecp** 管理命令和任务脚本时需要的一些语法规则和更重要的命令。

dcecp 命令语言由类似 **directory create** 和 **object modify** 的 DCE 管理命令，以及例如 **if** 和 **foreach** 的 Tcl 内置命令所组成。这里不讨论 DCE 管理命令，那些命令将在有关 DCE 组件管理的章节中讨论，这里讨论的焦点是更通用的语法规则和内置的命令。

dcecp 基于的工具命令语言 (Tcl) 是一种通用的语言，也可供 **dcecp** 以外的应用程序使用。虽然根据不同目的可以用多种方式使用 Tcl，但是我们的讨论仅限于管理 DCE 环境时最可能使用的命令。此外，我们对命令的讨论不会描述命令的每个方面，而是建议在管理 DCE 环境时为什么并且如何使用该命令。如果您对 Tcl 还不熟悉，则可能需要适当的、包含 Tcl 参考页的 Tcl 文档，以编写复杂的命令和任务脚本。

章节预览

本章逐个介绍 **dcecp** 的基本语法，然后说明在交互式任务和任务脚本中可能用到的一些命令。讨论将集中于:

- 使用变量作为将数据传递给命令或脚本的简单方法
- 将一个命令的输出作为另一个命令输入的方法，命令替换
- 将元素组合在一起以便 **dcecp** 正确地分析命令
- 使用列表来排序、查找并重用信息
- 在命令和任务脚本中使用算术函数
- 使用 **if** 语句和循环来执行和控制脚本
- 使用 **case** 命令根据关联的字符模式执行脚本
- 使用 **eval** 以合成命令
- 使用 **source** 导入操作
- 使用 **proc** 创建新的 **dcecp** 命令
- 使用出错和异常信息
- 处理字符串
- 使用文件
- 产生子进程

变量替换

类似其它的编程语言，**dcecp** 提供简写方式来表达和使用值。变量替换是一种简写方法，用于将一个值，比如在 CDS 目录（以一个变量）中对象的名称表示。

使用 **set** 命令可以为变量设置一个值。为了便于阅读，变量名可以由字母、数字和 _（下划线）组合而成。使用 『""』（引号）或 \（反斜杠）可以在变量名或值中包含空格（虽然通常不建议这么做）。下列示例都使用了有效的变量名:

```
set a $i
set CDS_clearinghouse_name cambridge_ch
set DCE_user_1 "William Rosenberry"
```

下列示例将变量 *a* 的值设置为 **7**。第二个示例使用不带值的 **set a** 命令使 **dcecp** 显示变量的当前值:

```
dcecp> set a 7
7
dcecp> set a
7
```

一旦使用 **dcecp set** 命令为变量设置了值，就可以在脚本或交互式命令的其他地方使用这个变量。DCE 控制程序使用 $（美圆符号）触发当前值在命令字中的插入，一个简单的示例如下:

```
dcecp> set a 7
7
dcecp> expr $a+2
9
```

这里首先将变量 *a* 设置为 7。第二行，使用 **expr** 命令对 *a* 的值(7)加2。美圆符号触发 **dcecp** 插入值 7。最后一行显示从 **expr** 命令返回的值。

一个更相应的示例如下:

```
dcecp> set a /.:/sec
/.:/sec
dcecp> object show $a
{RPC_ClassVersion
 {01 00}}
{RPC_ObjectUUIDs
 {06 3b 23 00 72 e5 e0 1d 8c b4 00 00 c0 8a df 56}}
{RPC_Group
 {2f 2e 2e 2e 2f 77 61 72 64 5f 63 65 6c 2e 6f 73 66 2e 6f 72
 67 2f 73 75 62 73 79 73 2f 64 63 65 2f 73 65 63 2f 6d 61 73 74
 65 72 00}}
{CDS_CTS 1994-05-23-17:21:37.481+00:00I0.000/00-00-c0-8a-df-56}
{CDS_UTS 1994-05-23-17:22:36.607+00:00I0.000/00-00-c0-8a-df-56}
{CDS_Class RPC_Group}
{CDS_ClassVersion 1.0}
dcecp>
```

使用 **unset** 命令除去（取消定义）一个变量，如示例:

```
dcecp> unset a
dcecp> set a
Error: cannot read "a": no such variable
dcecp>
```

命令替换

命令替换为在一个命令中表示另一个命令的返回值提供了一种简便的方法。在需要将一个命令的输出用作另一个命令的输入时非常有用。使用方括号来调用命令替换。下列示例使用了 **expr** 命令，我们将简单地讨论。通常 **expr** 执行一个数学函数，返回根据参数计算出的值，如下所示:

```
dcecp> set a 4
4
dcecp> set b [expr $a+2]
```

```
6
dcecp> set b
6
dcecp>
```

更实际的示例可能将命令替换用于一个返回长名称或列表的命令。让我们回忆一下第3页的『第1章 DCE 控制程序介绍』中的示例。在该示例中，**[group list temps]** 命令为 **foreach** 命令返回一个列表，对列表中的每个元素执行 **account modify** 操作。本节中，我们将进一步介绍 **foreach** 循环命令。

```
dcecp> foreach i [group list users] {
> account modify $i -change {expdate 1995-12-31}}
dcecp>
```

命令替换的另一个实际用途是为 **if** 语句设置测试条件。在第31页的『使用 if 语句判断条件』中展示了一个这种用法的示例。

组合元素并控制解释

编程语言通常使用某种符号，例如花括号、引号和括号，将所选择的元素作为整个组而不是个独立的元素进行操作。同样，**dcecp** 使用 『』（双引号）{}（花括号）将元素组成一个结构。双引号可以把一些通常被分别语法分析的元素组合成一个组，作为单一元素进行处理。花括号用来将元素组成一个列表，以便 **dcecp** 可以像分析返回值那样正确地分析命令和其它数据。

dcecp 命令的元素由空白符来分隔：空格、制表符和换行符。下列 **dcecp** 命令使用空格符来分隔它的三个元素：

```
dcecp> directory create /.:/subsys/comm_services
dcecp>
```

使用换行符或 ;（分号）分隔脚本中的命令。下列两个设置并使用变量的示例是等价的：

```
dcecp> set a /.:/subsys/comm_services
/.:/subsys/comm_services
dcecp> directory create $a
dcecp>
dcecp> set a /.:/subsys/comm_services; directory create $a
dcecp>
```

对于使用花括号或是引号来组合元素的选择取决于您希望 **dcecp** 如何解释如 $, [和 { 等特殊字符。花括号禁止对大多数这类字符的特殊解释，而双引号只禁止少数的特殊解释。反斜杠字符，在第22页的『用反斜杠包含特殊字符』中讨论，提供了另一种禁止解释特殊字符的方法。当同时使用时，花括号、引号和反斜杠为组合 **dcecp** 命令串提供了丰富的灵活性。

用花括号组合元素

花括号将分离的元素组合在一起以创建一个新的元素，新元素由 {（左花括号）和相应的 }（右花括号）之间的所有内容组成。可以嵌套用花括号括起的元素。下列每个示例的列表都包含三个元素：

```
larry moe curly
1 {3 5 7 11 13} {17 19}
red {orange yellow {green blue} indigo} violet
```

花括号禁止命令 ([ ]), 变量($)和反斜杠替换。花括号最重要的用途是确保 **dcecp** 命令有正确个数的参数，这也为在列表中包含特殊字符提供了一个便捷的方法。请看下列示例以了解其工作原理:

```
dcecp> set a solution
solution
dcecp> puts $a
solution
dcecp> puts {This is a convenient $a}
This is a convenient $a
```

虽然 **puts** 通常用来向文件中写入，但是如果在调用时只使用一个参数，则将参数输出到 **stdout**。在示例中，第一个 **puts** 允许对变量 *a* 的正常解释。第二个 **puts** 通过禁止对空格和美圆符的特殊解释而将分离的元素组合为一个参数。

用双引号组合元素

和花括号相似，双引号同样组合元素。不过不同的是双引号不能嵌套。而且，花括号几乎禁用所有的特殊字符，而双引号只禁用少数特殊字符: 空格、制表符、换行符和分号，这样可以避免在文本元素的字符串中使用棘手的反斜杠。双引号最方便之处就是使用 $ 符号触发器，以整洁、可读的方式进行变量的扩展。例如在下列示例中，为变量 (*a*) 设置包含空格的值:

```
dcecp> set a "XYZ server
for /.:/corp/comm_groups"
XYZ server for /.:/corp/comm_groups
dcecp> puts $a
XYZ server for /.:/corp/comm_groups
dcecp>
```

使用双引号不会禁止命令、变量和反斜杠的替换。请看第21页的『用花括号组合元素』中使用示例的变体:

```
dcecp> set
a solution
solution
dcecp> puts $a
solution
dcecp> puts "This is a convenient $a."
This is a convenient solution.
dcecp>
```

示例中，和第二个 **puts** 命令一起使用的双引号禁止对空格字符进行特殊解释，从而将五个元素组合为单一的变量传递给 **puts**。不过引号并不影响对美圆符进行解释。

用反斜杠包含特殊字符

我们已经知道 **dcecp** 依赖某些特殊字符，例如空格、花括号、引号或美圆符，来控制对元素的解释。有时可能需要在字符串中暂时停止解释而包含某个特殊字符。反斜杠提供了一种替代方式，可以用来抑制对紧随其后的字符的特殊解释。

可以用反斜杠在一个元素字符串中插入不可打印的间隔字符。例如，下列每个 **dcecp** 列表都有三个元素:

```
a b\ c d
a b \{
```

第一个示例中的元素为 **a**, **b c** 和 **d**。第二个示例中的元素为 **a**, **b** 和 **{**。更实际的示例可能使用反斜杠在错误消息中包含引号，如下列代码片段:

```
if {[llength $a] < 2} {
   error "Unable to parse \"$element_list\"."
}
```

下列列表给出了可以用反斜杠字符在元素字符串中包含的特殊字符:

\b	退格符
\t	制表符
\e	转义符
\n	换行符
\r	回车符
\{	左花括号
\}	右花括号
\[	左方括号
\]	右方括号
\$	$ 符号
\ （空格）	空格(" ")
\;	分号
\"	双引号
****	反斜杠
\（换行）	无
*****ddd*	八进制值

用注释为脚本编写文档

在编写脚本时，您可能想包含一些注释，以便提醒自己或其他人这一脚本所做的工作。使用 #（井号）可以插入注释。DCE 控制程序抑制对自井号至下一个换行符之间所有内容的解释。必须把井号放在 **dcecp** 期望出现的命令的第一个字符之前。下列两个示例都是有效的:

```
set a 5
# sets a to 5
set a 5 ;# sets a to 5
```

下列示例无效，因为井号没有出现在 **dcecp** 所期望的命令的第一个字符上。

```
set a 5 # sets a to 5
```

注释的通常用途是在脚本中为过程编写文档，如下列脚本样本片段:

```
#
# _dcp_cleanup_user_create  - This function undoes changes
# after a failure in one of the user create functions as
# though the operation never occurred.
#
proc _dcp_cleanup_user_create {account_name args}
{
```

便利变量

DCE 控制程序不仅记录输入的命令，而且记录命令的输出，并将信息的某些部分存储在便利变量中，供后续的命令再次使用。在交互式命令中使用这些变量可以减少输入并帮助消除输入错误。

便利变量只适用于 **dcecp** 命令，如 **directory**, **principal**, **acl**, **account** 等等。不适用于 Tcl 命令，例如 **for** 或 **eval**，也不适用于 UNIX 命令，例如 **mv** 或 **grep**。例如，便利变量 **_n** 存放在 **principal create** 操作中使用的名称（参数）。**principal show** 操作通过使用 **$_n** 变量检索这一名称。

```
dcecp> principal create D_Kalivas
dcecp> principal show $_n -all
{fullname {}}
{uid 17}
{uuid 00000011-d957-21cd-8d00-0000c08adf56}
{alias no}
{quota unlimited}
dcecp>
```

尽管这个简单的说明演示了便利变量的一般操作，不过这没有说明便利变量的益处。大部分的便利变量是供交互式使用的，不过有些也可以用在脚本中，这样可以提高灵活性，因为便利变量所包含的信息并不是硬编码在脚本中的。而且，当您使用 DCE 控制程序的经验丰富后，可能会发现这些变量是不可缺少的管理工具。

DCE 控制程序提供某些便利变量来替换一些原先输入的信息或命令的输出。所有便利变量以 _（下划线）开始，这样将一个字符的变量名保留做其他的用途。

下列部分描述便利变量。展示的顺序是将相似或相关变量排在一起。

当前主体（用户）名称 (_u)

_u 存放当前的简单主体名称。DCE 控制程序根据从父进程继承的登录上下文设置这个变量。通过执行另一个 **login** 操作可以更改这个值。使用 **set** 设置该变量时会产生错误。

```
dcecp> puts $_u
cell_admin
dcecp>
```

该变量的一种实际用途可以是在脚本中开始继续执行之前测试某个 DCE 身份。如果发现身份不正确，脚本可以提示输入必需的身份信息并执行一个 **dce_login** 操作。

如需有关构成全限定主体名称的信息，请参阅『当前单元名称 (_c)』中对单元名称变量的描述。

当前单元名称 (_c)

_c 便利变量存放主体所注册的单元名称。DCE 控制程序根据从父进程继承的登录上下文设置这个变量。通过执行另一个 **login** 操作可以更改这个值。使用 **set** 设置该变量时会产生错误。

```
dcecp> puts $_c
/.../my_cell.goodco.com
dcecp>
```

在管理员需要处理多个单元的环境中，这一变量相当有用。例如，可以将 **_c** 变量作为构件来构造当前上下文的全限定主体名称，以便在脚本中使用。如下列示例，将单元名和用户名变量用 /（斜杠）组合在一起:

```
dcecp> puts $_c/$_u
/.../my_cell.goodco.com/cell_admin
dcecp>
```

当前主机名称 (_h)

_h 便利变量存放当前主机的 DCE 名称。DCE 控制程序在调用 **dcecp** 时设置这个变量。使用 **set** 设置该变量时会产生错误。

```
dcecp> puts $_h
hosts/planets
dcecp>
```

_h 变量非常利于向交互式用户返回主机的名称。可以和 **_c** 变量一起使用来构造主机主体名称，如脚本所示:

```
dcecp> puts $_c/$_h/self
/.../my_cell.goodco.com/hosts/planets/self
dcecp>
```

最近操作参数名称 (_n)

_n 变量存放作为最近控制程序操作参数名称。大多数 **dcecp** 对象使用一个或者一列名称作为参数。那些不使用名称作为参数的命令包括杂项 **dcecp** 命令 **dcecp_initInterp**, **login**, **logout**, **errtext**, **quit**, **resolve** 和 **shell**。

名称通常是一个 **dcecp** 操作的第三个参数，如下列 **directory** 操作:

```
dcecp> directory create /.:/sales/printers/text_printers
dcecp>
```

一旦设置过，可以在后续操作的名称参数的位置使用 **$_n**。例如，可以修改在上一个示例中创建的目录 **/.:/sales/printers/text_printers** 的目录属性:

```
dcecp> directory mod $_n -change {CDS_Convergence
low}
dcecp>
```

在对多个名称执行目录服务操作时，**_n** 变量也可以存放一个名称列表。例如，可以创建几个目录，然后决定修改目录的一个属性:

```
dcecp> directory create {
> /.:/sales/printers/text_printers
> /.:/sales/printers/graphics_printers
> /.:/sales/printers/colorgraphics_printers }
dcecp>
```

后续的目录服务操作可以简单地使用 **_n** 变量作为名称或名称列表:

```
dcecp> directory
modify $_n -change {CDS_convergence high}
dcecp>
```

_n 的父代 (_p)

_p 变量含有存放在 **_n** 中名称的父代。**_n** 变量存放作为最近操作的参数的名称或名称的列表（见『最近操作参数名称 (_n)』）。**_p** 变量存放在层次上高于（更接近单元根目录）**_n** 中存放名称的名称或名称列表。

_p 变量的一个用途是向上遍历一个 CDS 分层目录。另一个用途是显示父代对象的*访问控制表*(ACL)。下列操作查看一个服务器配置对象及其父代对象的 ACL (**/.:/hosts/krypton/config/srvrconf**):

```
dcecp> acl show /.:/hosts/krypton/config/srvrconf/video_clip
{appl_admin cdfrwx}
{unauthenticated r}
{any_other r}
dcecp>
dcecp> puts $_p
/.:/hosts/krypton/config/srvrconf
dcecp>
dcecp> acl show $_p
{appl_admin criI}
{unauthenticated r}
{any_other r}
dcecp>
```

最近 dcecp 对象名称 (_o)

_o 变量存放在最近的操作所使用 **dcecp** 对象的名称。下列示例使用 **_o** 变量以避免重新输入 **account**:

```
dcecp> account show j_wanders
{acctvalid yes}
{client yes}
 .
 . [output omitted]
 .
{home /}
 .
 . [output omitted]
 .
{shell {}}
{stdtgtauth yes}
dcecp> $_o modify j_wanders -home /.:/fs/corporate_services/users/j_wanders
dcecp>
```

最近操作的返回值 (_r)

_r 变量存放最近一个操作的返回值。许多 **dcecp** 命令以列表形式返回多行的输出。

下列示例显示 **_r** 便利变量的一种用途。**dts show** 命令以列表形式返回多行。**attrlist getvalues** 操作（见 **attrlist(8dce)** 参考页）搜索整个列表以查找字符串 **toofewservers** 并返回所关联的值。

```
dcecp> dts show -counters
{creationtime 1994-09-16-07:50:13.067-04:00I-----}
{nointersections 0}
{nointersections 0}
{diffepochs 0}
{toofewservers 1}
{providertimeouts 82}
{badprotocols 0}
{badtimerep 0}
{noglobals 81}
{noresponses 0}
{abrupts 0}
{epochchanges 0}
{syserrors 0}
{syncs 1574}
{updates 0}
{enables 1}
{disables 0}
```

```
{nomemories 0}
{providerfailures 0}
{badlocalservers 0}
{badservers 0}
dcecp> attrlist getvalues $_r -type toofewservers
1
dcecp>
```

要使用的 DCE 服务器 (_s(xxx))

_s(*xxx***)** 变量存放用于下一个 DCE 操作的 DCE 服务器名称。DCE 控制程序提供四个这种类型的变量。因为 **dcecp** 不会设置这些变量，所以如果需要使用，用户必须自己设置这些变量。这些变量如下:

_s(sec)

这个变量存放在下一个注册表操作中需要使用的安全性服务器的名称。如果将其设置为一个只读副本，而操作（例如 **principal create** ）需要主副本，**dcecp** 将忽略该变量并尝试对主注册表进行绑定。使用 **_s(sec)** 变量的注册表操作包括 **principal**、**group**、**organization**、**registry**、**account** 和 **xattrschema**。

DCE 控制程序操作将 **_s(sec)** 变量和存放最近使用的注册表名称的 **_b(sec)** 变量一起混合使用。**registry** 操作采用下列次序来选择一个安全性服务器:

1. 使用作为名称参数传递给 **registry** 操作的服务器。
2. 如果该操作缺少一个名称参数，使用 **_s(sec)** 变量中指定的服务器。
3. 如果没有设置 **_s(sec)** 变量，使用 **_b(sec)** 变量中指定的服务器。
4. 如果没有设置 **_b(sec)**（即这是 **dcecp** 初始化以来的第一个 **registry** 操作），服务随机地为操作提供一个适用的服务器。

_s(cds)

这个变量存放需要在下一个目录服务操作中使用的 CDS 服务器的名称。如果设置过，CDS 会尝试使用所指定的服务器。如果尝试不成功，比如由于某种原因导致服务器不可用时，操作失败。要避免该故障，必须对这一变量 **unset** ，或者使服务器可用。

如果应用程序的所有要求都可以由变量中指定的中央交换库所满足，那么使用 **_s(cds)** 变量就很有意义。如果在 CDS 中的名称查找很可能遍历多个中央交换库中的目录时，建议不使用 **_s(cds)** 变量。在这种情况下，因为 **_s(cds)** 变量限制查找操作只使用指定的中央交换库，所以将产生查找错误。

_s(dts)

这个变量存放需要在下一个时间服务操作中使用的 DTS 服务器的名称。如果设置过，TDS 会尝试使用所指定的服务器。如果尝试不成功，比如由于某种原因导致服务器不可用时，操作失败。要避免该故障，必须对这一变量 **unset** ，或者使服务器可用。

该变量的一种用途是将 DTS 操作限制到单一的 DTS 服务器以达到监控的目的。一般情况下，时间服务可以使用任何可用的 DTS 服务器。

_s(aud)

这个变量存放需要在审计操作中使用的审计守护进程的名称。缺省情况下，审计操作只影响本地主机的守护进程。通过将 **_s(aud)** 变量的值指定为远程主机守护进程的名称，可以对远程主机的审计守护进程进行操作，如下所示:

```
dcecp> set _s(aud) /.:/hosts/planets/audit-server
/.:/hosts/planets/audit-server
dcecp>
```

如果设置了 **_s(aud)**，审计操作将尝试使用指定的守护进程。如果尝试不成功，比如由于某种原因导致审计守护进程不可用时，操作失败。要避免该故障，必须对 这一变量 **unset**，或者使审计守护进程可用。

可以按照下列格式指定一个 DCE 服务器或审计守护进程:

- 一个 DCE 名称。一个全局注册表名称的示例为 **/.../my_cell.goodco.com/subsys/dce/sec/oddball**。单元相关的 CDS 中央交换库名称的示例为 **/.:/Paris_CH**。
- 这一字符串绑定用来指定服务器所驻留的主机。字符串绑定可以表示安全性服务器、DTS 服务器与审计守护进程，而不能表示 CDS 服务器。一个字符串绑定的示例为 **{ncacn_ip_tcp 110.15.22.131}**。DCE 控制程序将绑定解析为主机上的适当服务。
- 单元名称，对于一个远程单元，指定一个全局的单元名称，例如 **/.../my_cell.goodco.com**。对于本地单元，可以用 **/.:** 指定根。这些操作使用一个适合于操作的任意服务器。

最近使用的安全性服务器 (_b(sec))

_b(sec) 便利变量存放最近 **registry** 操作所使用的安全性服务器的名称。DCE 控制程序根据前一个注册表操作设置这个变量。因此用户只能查看该变量，而不能进行设置。

读取这个变量值的一个原因是用来检查哪一个注册表执行了最近的操作，如下所示:

```
dcecp> puts $_b(sec)
/.../my_cell.goodco.com/subsys/dce/sec/oddball
dcecp>
```

注册表操作混合使用 **_b(sec)** 变量和 **_s(sec)** 变量来确定应当使用哪一个安全性服务器。参阅第27页的『要使用的 DCE 服务器 (_s(xxx))』 以了解 **_s(sec)** 变量，以及这些值是如何一起为注册表操作工作的。

最近错误代码 (_e)

_e 便利变量存放最近遇到的 DCE 错误代码。如果 DCE 控制程序能够确定错误代码的内容，就设置这一变量。如果实际的错误代码未知，这一变量被置为 **-1**（负一）。

CDS 信息级别 (_conf)

_conf 便利变量表示您在本地 CDS 守护进程中完成请求的信任度。它改变了大部分对 CDS 对象进行操作的命令的行为。信心级别可以是 **low**、**medium** 或 **high**。

使用表达式进行测量和计数

expr 命令为在脚本中表达和使用算术函数提供了灵活的方法。表达式非常利于比较数字的信息，例如列表元素中的个数，为监控目的设置阈值，递增控制脚本执行计数器的值，或者生成统计信息等等。

dcecp 简单的表达式是一个类似 **+**（加）或 *****（乘）这样的运算符和一些操作数的组合。**expr** 命令使用一个参数（表达式），因此如果表达式中带有空格就必须使用括号或花括号，可以用括号来控制表达式中的组合。表达式也可以嵌套。下列所有都是有效的表达式:

```
dcecp> expr {2 + 3}
5
dcecp> expr 2+3
5
dcecp> set x 24
24
dcecp> expr ($x-8)*2
32
dcecp> expr $x-(8*2)
8
dcecp> expr $x-8*2
8
dcecp>
```

在表达式中使用变量时必须小心，**$x** 这样的变量必须是如 24 这样的数字字符串，而不是的如 4*6 这样的非数字字符串。

DCE 控制程序通常将数字作为十进制整数进行处理，不过也可以按照八进制和十六进制格式来读取数字。在数字前加 0（零）表示八进制，例如 0477。在数字前加 0x 表示十六进制，例如 0x9FF。通过使用 ANSI C 标准中指定的格式（后缀 f、F、l 和 L 除外），可以用浮点格式表示数字。

通过访问 C math 库中的同名函数，DCE 控制程序在表达式中支持许多数学函数，例如 cos, exp, log, tan, sin 等等。

下列是可以和 **expr** 命令一起使用的运算符的部分列表。列表次序同样表示了优先次序。这表示，例如 **expr** 在加之前进行乘（2+2*4 等于 10）。

-	一元减
~	按位否
!	逻辑非
*****	乘
/	除
%	取余
+	加
-	减
<<	向左移位
>>	向右移位
<	布尔小于
<=	布尔小于或等于
>	布尔大于
>=	布尔大于或等于
==	布尔等于
!=	不等于
&	按位与
^	按位异或

|	按位或
&&	逻辑与
||	逻辑或
*a***?** *b***:** *c*	if-then-else （同 C 语言）。

对列表操作

列表提供了对一组元素，例如主体集合、组成员或其它对象，进行操作的简便方法。列表是一组由用户输入或从命令返回的对象的集合。在本章以前的示例中，我们已经看到了列表，它们是由空格、制表符和换行符分隔的任意个数的元素。列表通常用花括号括起。

下列都是列表的示例:

```
{n_long l_jones p_sawyer d_witt m_dougherty s_preska}
{{7.:/hosts} {/.:/subsys}}
```

DCE 控制程序使用列表对元素进行组合，以便 **dcecp** 命令解释器正确地进行分析。例如，set 命令使用两个参数:

```
set varName value
```

下列 **set** 不能被正确分析，因为 **dcecp** 检测到第三个参数:

```
dcecp> set a John Hunter
Error: wrong # args: should be "set varName ?newValue?"
dcecp>
```

使用花括号、引号或反斜杠来创建有效的列表，如下:

```
dcecp> set a {John Hunter}
John Hunter
dcecp> set a "John Hunter"
John Hunter
dcecp> set a John\ Hunter
John Hunter
dcecp>
```

对列表进行操作的命令提供了一种简便方法，用来对列表中元素单独或整体地进行求值、选择和操作。DCE 控制程序提供了一组广泛的命令以创建、修改、搜索、排序或转换列表。

例如，下列脚本返回列表中最后一个元素。**llength** 命令返回列表中元素的个数。列表中有四个元素，所以 **llength** 返回 **4**。DCE 控制程序从 **0**（零）开始自左向右对元素编号，所以有三个元素的列表带有编号为 **0**, **1**, **2** 和 **3** 的元素。变量 *c* 的值被设置为列表中最后一个元素的编号 (3)。最后，**lindex** 命令返回 2 号元素 (**f**)。

```
dcecp> set a {a b {c
d e} f}
a b {c d e} f
dcecp> set b [llength $a]
4
dcecp> set c [expr $b-1]
3
dcecp> lindex $a $c
f
dcecp>
```

DCE 控制程序提供了许多对列表操作的命令。可以用 **concat** 命令将列表合并到一起。用 **linsert** 向现存的列表中添加元素。用 **lrange** 抽取某个范围的元素，用 **lreplace** 替换列表中的元素，或用 **lsort** 将列表元素按照字典次序排序。DCE 控制程序还包含一个用于操纵列表元素的 **attrlist** 对象（请参阅 **attrlist(8dce)**）。

这里有一个按照字母顺序显示树中所有子目录的示例。变量 **_r** 是存放最近命令输出的 **dcecp** 便利变量。示例中，**_r** 存放由 **directory list -simple** 命令返回的目录列表。

```
dcecp> directory list -simple /.:
hosts subsys cell-profile fs lan-profile planets_ch sec sec-v1
dcecp> lsort $_r
cell-profile fs hosts lan-profile planets_ch sec sec-v1 subsys
dcecp>
```

控制脚本

DCE 控制程序提供了几个用于控制脚本执行的命令。例如 **if**, **while**, **for**, **foreach** 和 **case**，这些命令在不同情况下执行脚本的某部分。**break** 和 **continue** 命令可以停止部分或整个脚本的执行。

使用 if 语句判断条件

有时，您可能希望脚本的一部分仅在某种条件下才执行。使用 **if** 语句可以检测一个条件，并有条件地执行某个操作。**if** 语句的语法为:

```
if test true_body else false_body
```

假设正在编写一个对一列属性进行搜索以查找某个特定属性的脚本。一个 **if** 语句可以根据一个属性是否存在而采取特定的操作。在下列的示例脚本中，如果帐户名称没有在 **list_of_group_entries** 变量中出现，将产生一个错误消息:

```
set list_of_group_entries [group list $group -simplename]
if { [lsearch $list_of_group_entries $account_name] == -1} {
     group add $group -member $account_name
           } else {
error "Group \"$group\" already has an entry \
  for \"$account_name\"."
}
```

使用循环控制脚本执行

编程语言使用循环在指定条件成立的情况下重复操作。DCE 控制程序提供三种类型的循环: **foreach**, **while** 和 **for**。所使用的循环类型取决于条件的指定方式。

foreach 循环

当您需要对列表中每个元素执行一个指定操作时，使用 **foreach** 命令。请记住，列表是您输入或从命令返回的一组对象或其他内容。

语法格式为:

```
foreach variable_name list body
```

foreach 由列表，脚本体和按次序表示列表中每个元素的变量所组成。这一命令运行由变量表示的元素脚本体，并将变量设置为列表中的下一个元素。

下列样本 **foreach** 命令可能是用于管理 DCE 单元中主机的脚本程序的一部分。如果在配置单元中某个主机时发生故障，在这一脚本片断则从注册表中删除主机主体名。

foreach 命令察看单元中每个主体的名称。如果 **string** 命令在 **principal catalog** 的输出中找到了主机名，则将主体名称从注册表中删除。

```
foreach princ [principal catalog -simplename] {
    if {[string match $host_name [string range $princ 0 \
        [expr [string length $host_name] - 1]]] == 1} {
        principal delete $princ
    }
}
```

请记住，循环将其结果返回到解释器而不是 **stdout**。必须采取额外的步骤才能将结果发送到 **stdout**。下一个示例使用 **puts** 命令将 **foreach** 的结果发送到 **stdout**:

```
foreach i [group list subsys/dce/dts-servers] {
    puts [principal show $i]
}
```

在脚本中，也可以用 **append** 命令将所有结果附加到一个变量中，或者用 **lappend** 将结果作为列表元素进行附加，如下:

```
foreach i [group list subsys/dce/dts-servers] {
  append result [principal show $i]
}
return $result
```

while 循环

while 循环和 C 语言中的 **while** 循环相似。它使用两个参数: 表达式和脚本（称为*脚本体*）。如果表达式的值非零，**while** 执行脚本体，并且对表达式重新求值，继续循环直到表达式的值为 0。**while** 循环的语法为:

while *expression body*

下列示例过程使用 **while** 循环搜索列表中的每个元素以查找模式。只要列表的长度大于 0 (**$size > 0**)，过程继续循环。

```
proc _dcp_list_find {search_list pattern} {

  set found_items ""
  set size [llength $search_list]
  while { $size > 0 } {
    set size [expr $size - 1]
    set index [lsearch $search_list $pattern]
    if { $index == -1 } {
      return $found_items
    }
    lappend found_items [lindex $search_list $index]
    set search_list [lreplace $search_list $index $index]
  }
}
```

for 循环

for 循环同样和 C 语言中的相似。虽然 **for** 比 **while** 更复杂，但是 **for** 将所有循环控制信息保持在一起，从而更容易理解。**for** 命令的语法为:

for *initial_expression test reinit script_body*

要使用 **for**，需要设置一个初始表达式，然后执行脚本体之前测试条件。在执行完脚本体后，**for** 命令对初始表达式重新初始化，并对新的值再次测试，重复执行循环直到测试的结果为假。

下列示例显示 **for** 循环将某个操作执行一定次数，然后停止。这个示例在注册表中创建 50 个 guest 主体名称。

```
dcecp> for {set i 0} {$i < 50}
{incr i} {
> principal create guest$i
> }
dcecp>
```

使用 continue 和 break 终止循环

continue 和 **break** 命令终止由 **while**, **for** 和 **foreach** 命令开始的循环。

使用 **continue** 命令终止循环的当前迭代。例如，在对其余元素继续操作的同时，循环可以测试并有选择地忽略列表中的特定元素。使用 **break** 命令立即终止循环的执行。

下列示例脚本片断是一个包含 **continue** 和 **break** 命令的 **foreach** 命令循环。**foreach** 命令查看一个单元中所有的 DTS 服务器，直到找到一个时间提供者。（时间提供者是一个特殊的 DTS 服务器，从外部的时间源接收时间。）如果列表（由 **dts catalog** 操作创建）中的第一个服务器在 **dts show** 操作中返回输出，**continue** 命令调用脚本中的下一行，在输出中搜索 **{provider yes}** 属性和值。如果 **provider** 属性（由 **attrlist getval** 操作检查）为 **yes**，脚本将 **server** 变量设置为该 DTS 服务器的名称，然后 **break** 命令终止整个 **foreach** 循环。

```
foreach s [dts catalog] {
    if {[catch {dts show $s} dts_sh_out] != 0} {
         continue
    }
    set p [attrlist getval $dts_sh_out -type provider]
    if {[string match $p "yes"] == 1} {
        set provider "yes"
        set server $s
        break
    }
set provider "no"
}
```

使用 case 在执行前测试模式

某些命令返回一个列表，例如一个目录中对象的列表，或者运行在一个主机系统上的服务器的列表。可以用 **case** 命令来测试一个列表或字符串以查找指定的模式，例如特定对象或服务器的名称。在检测到指定的模式后，**case** 命令执行和该模式关联的脚本。**case** 命令的语法为:

case *string* **in** *pattern* {*script*} *pattern* {*script*}

case 命令在 *string* 中查找 *pattern*，并执行 {*script*}。单词 **in** 可以省略。下列示例说明 **case** 命令如何工作:

```
dcecp> set x {one ten twenty}
one ten twenty
dcecp> foreach el $x {case $el in one {puts script1} two {puts script2}}
script1
dcecp>
```

case 命令首先在 **$x** 中查找模式 **one**。如果找到该模式，相应的脚本在屏幕上显示 **script 1**。如果找不到更多的匹配，**case** 命令结束。

对于更实际的示例，假设您运行一个列出特定系统中所有服务器的 **dcecp** 命令。可以在列表中搜索特定的服务器名，并且执行一个将每个名称附加到某个特定文件的脚本，如下:

```
case $x in server1 {lappend filename1} server2 {lappend filename2}
```

如果模式的列表很长而且可能跨越多行，可以把目标模式和脚本用花括号括起，以避免将换行符作为分隔符进行解释。这样还有额外的好处，即防止在花括号列表中出现变量和命令替换。

模式可以包含通配符。搜索模式中的 ?（问号）匹配目标模式中的任何单一字符。例如，**?at** 匹配 **bat** 和 **hat**。模式中的 *（星号）匹配目标模式中的任何字符串。例如，***at** 匹配 **bat** 和 ″**three cornered hat**″（ 注意使用引号禁止将空格作为分隔符解释的方法)。

您可能需要一种在没有找到匹配的模式时执行脚本的方法。**case** 命令有一个特殊的模式，称为 *default*，其相应的脚本在没有查找到匹配的模式时执行。应当将 default 模式放在列表的最后位置:

```
case $x in {
  a {puts "script for case a"}
  b {puts "script for case b"}
  default {puts "run this script if no matches are found"}
}
```

动态创建命令

eval 命令允许您将较小的脚本链接在一起以创建新脚本。这一技术可以用于在脚本中记录管理员对各种问题的响应，然后根据响应构造一个专门的脚本。语法格式为:

eval *arg ...arg*

下列示例使用变量保存用于 **account create** 操作的值和选项。 **eval** 命令确保变量被正确地扩充并执行。

```
dcecp> set
mpwd {-mypwd mxyzptlk}
-mypwd mxyzptlk
dcecp> set pwd {-password change.me}
-password change.me
dcecp> set org {-organization guests}
-organization guests
dcecp> set grp {-group guest}
-group guest
dcecp> eval account create guest1 $mpwd $pwd $org $grp
dcecp>
```

在使用变量构造 **eval** 命令时必须小心，类似下列的 **eval** 命令有时会出现问题，因为 **dcecp** 对其进行了两次语法分析。首先，**dcecp** 对 **eval** 命令及其参数进行语法分析，然后在按脚本执行时对 **eval** 参数进行语法分析。

```
dcecp> eval $a $b $c
dcecp>
```

在参数前后使用花括号可以避免一些语法分析问题，如下:

```
dcecp> eval {$a $b $c}
dcecp>
```

为了保证 **dcecp** 对 **eval** 命令进行正确语法分析，可以调用 **dcecp list** 命令以生成一个有效的列表结构:

```
dcecp> eval [list $a $b $c]
dcecp>
```

将其它文件作为 dcecp 脚本读取

source 命令读取其它文件的内容，并将内容作为 **dcecp** 脚本执行。这一能力可以让您将低层脚本组合在一起构造成高层脚本，就像搭积木。因为您重用脚本而不是直接复制并稍加修改，这样脚本将更一致，而且易于开发和维护。语法格式为:

source *filename*

source 的返回值是 *filename* 中最后一个命令的返回值。

作为实际的示例，假设已经有一个列出 CDS 子树中项的脚本，一个删除子树的脚本和一个移动子树的脚本。这些脚本共同需要的函数可能是列出子树根下所有子目录的函数。可以编写一个列出每个子目录的脚本，并将其命名为 **children_list.dcp**。（**.dcp**扩展名是 **dcecp** 命名脚本文件的约定。）当任何脚本需要列出所有子目录时，直接使用 **source** 命令:

```
source children_list.dcp
```

使用 **return** 命令终止一个 **source** 命令。**return** 命令为 **source** 和 **proc** 命令提供了即使发生正常或异常的错误情况下仍然能够以受控方式退出的方法。**return** 命令管理出错信息并继续执行，而不是让错误情况导致整个脚本退出。我们将在第39页的『处理错误和异常』中和其它错误处理技术一起讨论 **return** 的用法。

创建新命令

DCE 为控制和监控操作提供了的功能强大的命令集用。但是由于难以预测最终用户对 DCE 的具体使用，所以有些管理员需要附加的命令来完成非常特定的要求。**proc** 命令为创建附加的命令提供简易的方法，使这些命令的外观和工作方式都和内置命令相似，例如 **set**, **list** 和 **while**。不过内置命令是用 C 语言编写的，由 **proc** 创建的命令是用脚本编写的，如下:

```
dcecp> proc div {x y} {expr $x/$y}
dcecp>
```

proc 命令使用三个参数: 过程名称，过程参数名称的列表和构成新的过程体的 **dcecp** 脚本。新过程 **div** 需要两个参数，例如:

```
dcecp> div 12 4
3
dcecp>
```

缺省情况下，**proc** 假设所有变量都是局部变量。即变量的名称和值都只在过程中设置，在过程完成后自然失效。下列命令产生一个错误，因为没有在过程中设置变量 *x* 和 *y*:

```
dcecp> set x 15
15
dcecp> set y 3
3
dcecp> proc div {} {expr $x/$y}
dcecp> div
Error: cannot read "x": no such variable
```

可以使用 **global** 命令导入全局变量（在过程外定义的变量）：

```
dcecp> set
x 15
dcecp> set y 3
dcecp> proc div {} {
> global x y
> expr $x/$y
> }
dcecp> div
5
dcecp>
```

一旦导入一个全局变量，它在过程的持续时间内始终有效。过程可以使用 **unset** 和 **set** 来更改变量的值。新的值在过程的内部和外部都可以使用，如上所示。

使用 **return** 命令可以使过程立即返回。**return** 参数的值就是过程的返回值。

```
proc find {a} {
 <some pattern matching script that looks for a specific CDS entry>
    if {a != b} {
       return 1
    }
    return 0
}
```

可以设计过程不使用参数或者使用可变个数的参数。例如，不带参数的过程可以简单地执行一些直接的操作，如下列示例:

```
proc _do_create_group {} {
     global rpcgroupname
     rpcgroup create $rpcgroupname
}
```

在参数列表中使用嵌套式列表结构可以为参数指定缺省值。下列示例中，必须提供第一个参数 *attr*，如果没有提供第二个参数 *value*，则缺省为 **unset**。

```
proc _attr_show {attr {value "unset"}}  {
     puts "$attr is $value"
}
```

过程可以调用其它过程。当前过程可以使用 **upvar** 命令从调用该过程的过程中导入变量，如下:

```
upvar level otherVar1 myVar1 otherVar2 myVar2
```

如果 *level* 参数为 **1** 则获取父过程的变量上下文，如果参数为 **2** 则获取父过程的过程的变量上下文。在 *level* 参数前加 **#** 可以指定相对于全局上下文的级别。*level* **#0** 获取全局变量。*level* **#1** 从自全局级别调用的过程中获取变量。

otherVar 参数指定要导入的变量。需要包含 **myVar** 参数来将变量重命名，以便在当前过程中使用。下列示例将导入的变量重命名为 **cargs**:

```
upvar 1 local_args cargs
```

通过使用 **uplevel** 命令，过程可以执行父过程上下文中的脚本。这一命令为管理过程的上下文提供了简便的方法。例如，使用 **uplevel** 可以直接连接所有父过程中的变量，而不必逐个导入或操纵那些变量。语法格式为:

```
uplevel level arg arg arg
```

uplevel 命令和 **eval** 相似，将参数并置并作为脚本执行，不过和 **eval** 不同的是 **uplevel** 在由 *level* 指定的上下文中执行，而不是在当前上下文中执行。*level* 参数在 **uplevel** 中的工作方式和 **upvar** 中相同。*level* 参数为 **1**，使用父过程的上下文。*level* 参数为 **#1**，使用第一级上下文。

如果 **proc** 命令指定了一个已经有效的命令名，那么新的过程替换现存的同名过程。除非特别需要，不应命名新的命令以避免替换现存的内置命令。

使用 **rename** 命令可以重命名或删除 Tcl 命令。例如，可以暂时将 **list** 重命名为 **list.old**，然后使用 **proc** 创建另一个称为 **list** 的命令。在使用完制造的 **list** 命令之后，可以将 **list.old** 重命名为 **list**，以恢复 **list** 的原始功能，如下所示:

```
rename list list.old
proc list {} {
     <some list operation>
}
rename list.old list
```

通过省略 **rename** 命令的第二个参数可以删除一个命令。下列示例删除 **list** 命令:

```
rename list
```

字符串处理

许多 DCE 管理的操作返回某种类型的信息。例如，**principal show** 返回关于主体的信息。通常这些信息的格式为列表，如下例:

```
dcecp> principal show R_Parsons
{fullname {}}
{uid 15}
{uuid 0000000f-d6f9-21cd-8d00-0000c08adf56}
{alias no}
{quota unlimited}
{groups users}
dcecp>
```

虽然对管理员来说，扫描列表并抽取必需的信息很容易，但是对脚本的操作不同。脚本在搜索特定信息时不使用列表的概念，而是对组成列表的字符集合（称为**字符串**）进行操作。DCE 控制程序提供一组命令对字符串进行操作，用于构造、分析、比较、修改字符串，或者从字符串中抽取值。

构造字符串

脚本经常需要为其它命令或在屏幕上显示来构造字符串。DCE 控制程序提供了 **format** 命令以便在脚本中构造字符串。

format 命令根据需要替换变量。下列示例用 **format** 命令将单元名称字符串（**字符串**类型由 **%s**表示）附加到字符串 **/hosts** 之前，以构造变量 **_dcp_host_entries**。单元名称包含在 **_c** 便利变量中。

```
dcecp> set _dcp_host_entries [format "%s/hosts" $_c]
/.../my_cell.goodco.com/hosts
dcecp>
```

format 命令同样可以将参数在十进制、八进制、十六进制、浮点和科学记数法等格式之间进行转换。可以指定打印或省略正负号，左对齐或右对齐输出，填充空格或零。下列示例将整数 8 转换为八进制的等价值。第二个示例将输出右移九个空格。

```
dcecp> format
%1o 8
10
dcecp> format %9o 8
       10
dcecp>
```

语法分析字符串语法

DCE 控制程序包含一个 **scan** 命令，用于对字符串进行语法分析，对适当的部分进行转换并存储在变量中。这一功能很有用，比如在将前一个命令返回的信息转换为可以用于另一个命令输入的数据时。**scan** 命令的语法为:

scan "*string*" "*format*" [*varname* [*varname*]...]

可以用文字或变量指定 *string*。*format* 部分控制语法分析，忽略为了提高可读性在 *format* 部分中包含的空白和制表符。该部分由一个或多个以 %（百分号）分隔的转换指示符组成。转换指示符定义 *string* 的哪个部分被转换和存储，并定义转换的类型。

下列示例分析包含在变量 **_dcp_temp** 中的字符串，查找一个有效的浮点数，并将浮点数存储在变量 **_dcp_temp2** 中:

```
if { [scan $_dcp_temp "%f" _dcp_temp2] != 1 } {
   error "Variable \"$_dcp_temp\" is not a \
     valid floating-point number"
}
```

其它字符串处理操作

使用 **string index** 和 **string range** 可以指定字符串中的某个字符或某个范围内的字符。这些命令用来从字符串中抽取可预估长度的信息。

string index 命令使用一个参数，该参数表示需要从字符串中抽取的字符的位置（自左向右从 0（零）开始）。**string range** 命令包含两个参数，表示包含在范围中的最左和最右字符的位置。下列示例说明 **string range** 命令的一种用法:

```
dcecp> string range {The
quick brown fox} 4 9
quick
dcecp>
```

使用 **string compare** 可以按照字典顺序（字母次序）比较一个字符串是否大于、小于或等于另一个字符串。通常，这个运算对组成字符串的 ASCII 码进行字节比较。

使用 **string length** 命令统计字符串中字符的个数。例如:

```
dcecp> string length "The
quick brown fox"
19
dcecp>
```

使用 **string toupper** 和 **string tolower** 命令将字符在大写和小写之间转换。例如:

```
dcecp> string toupper "The quick brown fox"
THE QUICK BROWN FOX
dcecp>
```

使用 **string trim** 命令从一个字符串中除去特定字符。使用 **string trimleft** 和 **string trimright** 命令除去字符串中最左或最右的字符。

可以用多种方式执行模式匹配。使用 **string match** 命令调用 『glob』 风格的模式匹配。这模仿 **csh** 中的 glob 模式匹配，返回 **1** 表示匹配，**0** 表示不匹配。可以使用 **regexp** 命令执行更灵活的正则表达式模式匹配（如同 **egrep** 中的那样）。使用 **regsub** 命令可以将操作扩展为执行正则表达式替换。

下列示例说明 **regsub** 命令的用法。第一个参数指定搜索模式，第二个参数是搜索的字符串，第三个参数指定替换模式，最后一个参数是 **regsub** 存放新字符串的变量。如果没有发生替换，这一命令返回 **0**，如果发生替换则返回 **1**。

```
dcecp> regsub brown
"The quick brown fox" blue color
1
dcecp> puts $color
The quick blue fox
```

处理错误和异常

dcecp 解释器包含错误处理功能，在 **dcecp** 脚本出错时返回错误信息。错误信息可以告诉用户什么出错，这样可以避免再犯相同的错误。导致 **dcecp** 错误的原因很多，例如，命令没有收到正确数目的参数，命令可能有某种外形错误，以及因为某种原因导致一个操作对象（比如一个 CDS 目录）不可用。

这里讨论处理错误和异常的三种方法:

- 使用全局错误信息变量
- 捕获异常
- 重发复杂错误

使用全局错误信息变量

dcecp 在遇到错误时打印一个描述性消息，例如:

```
Error: wrong # args: should be "set varName ?newValue?"
```

有些情况下，仅凭错误消息难以确定问题发生的准确位置。因此 **dcecp** 在一个叫做 **errorInfo** 的全局变量中存储附加的错误信息。脚本可以访问并打印这些信息来帮助查找错误。通常在发生错误时，它跟踪正在执行的命令。

下列示例显示可以存储在 **errorInfo** 中的信息。反过来读，您可以确定错误发生在 **user** 操作中 **_dcp_create_user** 过程所调用的 **parseargsuser** 过程的脚本体的第四行附近。

```
dcecp> puts $errorInfo
Unknown option "group"
    while executing
"
    invoked from within
"
    ("while" body line 4)
    invoked from within
"
    (procedure "parseargs" line 60)
    invoked from within
"
    (procedure "_dcp_create_user" line 64)
    invoked from within
"
    invoked from within
"
    invoked from within
```

```
"
    (procedure "user" line 24)
"
dcecp>
```

另外，**dcecp** 可能在叫做 **errorCode** 的全局变量中存储另一种错误信息。这一变量包含类似下列的列表，可以用来标识其它类别的错误。

```
UNIX, ENOENT, "insufficient arguments for filename"
```

如果一个错误产生无用的错误信息，DCE 控制程序将 **errorCode** 变量设置为 NONE。

使用 catch 捕获错误和异常

有时候需要捕获某种类型的错误，而不是让错误终止活动的命令。**catch** 命令允许脚本捕获并忽略错误以便继续处理。假设脚本要重命名一个命令，如果该命令存在的话。然而，在执行 **rename** 命令时该命令名可能不存在。

```
dcecp> rename move move.old
Error: cannot rename "move":  command does not exist
dcecp>
```

使用 **catch** 将 **rename** 命令作为脚本调用。

```
dcecp> catch {ren move move.old}
1
dcecp>
```

catch 将参数作为脚本处理并执行，执行成功则返回 **0**，如果发生错误，则由 **catch** 捕获并返回 **1**。

可以对 **catch** 命令添加第二个参数，这个参数是一个变量， **catch** 在其中存放返回值（如果成功完成）或错误消息。**catch** 命令的语法为:

catch *command varName*

catch 在脚本中的一种用途是用于调用其它过程。可以将下列脚本片断这样理解: 『如果 **_dcp_create_group** 过程返回不成功(**!= 0**) 则执行 **_dcp_cleanup_user_create** 过程并显示存储在 **msg** 变量中的错误。』

```
if {[ catch {_dcp_create_group $group group_created} msg] != 0 } {
      _dcp_cleanup_user_create $element -principal
      error $msg
}
```

异常是由 **break**, **continue** 和 **return** 命令生成的一类特殊的错误。可以用 **break** 和 **continue** 命令中断诸如 **while**, **for** 和 **foreach** 的循环，用 **return** 命令终止一个 **proc** 或 **source** 命令。

如果在一个更综合的命令中使用循环，那么过程就更难处理异常结果。例如，一个用户编写的在 CDS 中搜索特定对象类型的过程可能调用 **foreach** 来测试特定属性是否存在。

如果使用 **break**, **continue** 或 **return** 命令来管理循环的执行或者其它嵌套的命令（例如 **case** 或 **if**），父命令就很难捕获异常。父命令将终止并发出错误消息。然而，由于错误是和父命令关联的，所以很难跟踪到实际发生错误的循环命令。

如果必须用 **continue**、**break** 或 **return** 来终止一个被另一个命令所调用的命令，请考虑使用 **catch** 来调用嵌套的命令，由它来调用相应的 **continue**, **break** 或 **return** 命令，以便从错误和异常中恢复。采用这个方法，**catch** 命令将异常保持在循环或嵌套的过程之内，这样就很容易跟踪错误。

```
foreach s [server catalog] {
    if {[catch {server show $s} srv_sh_out] != 0} {
         continue
    }
```

重新发出复杂错误

proc 命令允许您创建执行非常严格操作的过程或命令。例如，叫做 **_dcp_get_servers** 的过程检索并过滤正在运行的服务器的信息，它可能包含*嵌套的*命令或过程以执行各种子任务，例如在服务器信息中循环查找某个字符串。虽然使用嵌套的命令或过程可以开发出更综合的过程或命令，但是如果没有适当地对错误进行传递，将很难定位所产生的错误。

复杂的脚本可以使用 **error** 命令来重发由脚本先前部分触发的错误。下列脚本片断只是简单地打印出硬编码的错误消息。这一方式可以用来定制剪裁过的信息，以详细说明错误情况。

```
set dts_cat_out [_dcp_dts_catalog]
if {[llength $dts_cat_out] == 0} {
    error "Unable to find any DTS servers"
}
```

下一个脚本片断则做的更多，它使用 **catch** 将任何从 **_dcp_create_group** 过程返回的错误信息存储到 **msg** 变量中。如果发生故障 (**!= 0**)，脚本调用一个清除过程，撤销所做的一切工作，然后打印出 **msg** 变量中存储的错误消息。

```
if {[ catch {_dcp_create_group $group group_created} msg] != 0 } {
      _dcp_cleanup_user_create $element -principal
      error $msg
}
```

以上讨论提供了一些相当简单的错误处理技术。注意，错误处理可能很复杂，尤其是在更复杂的情况下。建议您阅读其它详细介绍 Tcl 用法的出版物，以更多地了解错误处理。

使用文件

DCE 控制程序有几个用于读取和写入文件的命令。文件可以用于存储 **dcecp** 输出的操作供以后参考。这里是几个有用的文件操作示例:

- 可以对一个单元中所有主机运行一个 **server catalog** 操作，并将来自每个主机的结果存储在和主机相关的文件中。随后可以比较这些文件并生成一个关于服务器配置的报告。
- 通过运行一个显示每个帐户最近登录时间的 **dcecp** 脚本，可以检测非活动的帐户，并将这些信息存储在文件中供将来评估。
- 也可以修改难以用 **dcecp hostdata** 对象操纵的 DCE 文件。例如，可以编写一个为 **cds_attributes** 文件添加新属性的功能。

OSF 提供的 DCE 目前只为 UNIX 文件或支持 POSIX 系统调用的系统提供文件操作。不过一些供应商的 DCE 版本可能支持其它系统上的文件操作。

指定文件名

使用习惯的 UNIX 规则指定文件名。例如，**/opt/dcelocal/dcecp/server_snap.dcecp** 指定在 **/opt/dcelocal/dcecp** 目录下叫做 **server_snap.dcecp** 的文件。也可以用相对文件名指定文件，例如 **~dce_admin/scripts/server_snap.dcecp** 和 **~/admin/server_snap.dcecp**。可以使用 **pwd** 命令打印当前工作目录，使用 **cd** 命令设置当前工作目录。下列命令将当前目录设置为 **~dce_admin/scripts**:

```
dcecp> cd ~dce_admin/scripts
dcecp>
```

可以使用 **glob** 命令查看一个目录下的文件列表。这个命令返回匹配参数中指定模式的文件名列表。例如:

```
dcecp> glob *
help local_lib.dcp
dcecp>
```

可以用 **file** 命令及各种选项来查看文件的其它信息。**file** 命令可以根据文件的时间、大小或许可权（当前用户是否可以执行、读取或写入）选择文件。

读写文件

dcecp 中用于读写文件的命令和 C 语言中的对应部分类似，例如 **fopen**, **fclose** 等等。

使用 **open** 命令为读写打开一个文件。**open** 命令的第二个参数指定文件的访问方式（如下列示例中的 **+r**）。可以为读或写或者同时读写打开文件，并且指定是用新的信息替换现存文件还是向其中添加新的信息。可以将初始的位置设置为文件的开始或结束。缺省的访问方式是只读（文件必须存在）。

```
dcecp> open server_snap.dcecp
+r
file5
dcecp>
```

open 在打开每个文件后为其分配一个文件标识符。在随后的命令中使用文件标识符引用文件。

文件一旦打开，可以使用 **puts** 命令向其中添加行。一般情况下，**dcecp** 只有在累积了足够的信息后才向文件中写入。如果需要 **dcecp** 立即将信息写入文件，使用 **flush** 命令。使用 **gets** 从文件中获取下一行，或者用 **read** 在文件中读取一定字节的内容或所有内容。下列示例在名为 **prins** 的文件中写入所有主体的列表:

```
dcecp> open prins w+
file8
dcecp> puts file8 [principal catalog]
dcecp> close file8
dcecp>
```

有时不需要从文件的第一行开始读写。DCE 控制程序提供几个设置访问位置的程序，这样就不必在文件中逐行地前进了。如果对终端或其它不支持随机访问的连续设备使用这些命令则将产生错误。使用 **seek** 命令设置文件的访问点。将偏移量指定为从起始地址移动的字节数，起始地址可以是文件的开始位置、结束位置或当前位置。使用负数表示向文件的开始位置移动，例如下列示例中从当前访问位置前移 16 个字节。

```
dcecp> seek file5
-16 current
dcecp>
```

使用 **tell** 命令可确定当前访问位置。将返回值保存到变量中，这样随后可以再返回到该位置。

最后，可以使用 **close** 命令关闭文件，如下:

```
dcecp> close file5
dcecp>
```

产生子进程

使用子进程执行命令为一些复杂的脚本或特殊的管理要求提供了多种解决方案。子进程可以提供:

- 对操作系统命令的访问
- 一种建立同步、顺序执行的方式
- 简化复杂或高级脚本的方法

从脚本中运行操作系统命令

虽然 DCE 控制程序功能强大，但是有时脚本需要使用操作系统命令来完成一些简单的（甚至不简单的）操作。**exec** 命令通过创建子进程为脚本提供了执行外部命令的方法。下列示例使用 **exec** 命令检索本地主机名称，然后存放在 *hostname* 变量中，供随后使用。

```
dcecp> set hostname
[exec hostname]
myhost
dcecp> directory list /.:/hosts/$hostname -simple
cds-clerk cds-server dts-entity profile self
dcecp>
```

exec 命令通常返回子进程操作的执行结果。不过可以使用 UNIX 重定向符号 (<, << 和 >) 将标准输入或标准输出重定向。也可以用 | (垂直条) 将输出通过过滤器，例如 **nroff**, **sort** 或 **grep** 进行连接。

在单独使用时，**exec** 命令是同步的，这表示外部命令在脚本继续执行之前完成，不过如果子进程需要很长时间才完成，例如同步一个 CDS 单元中的目录，可以在 **exec** 命令中使用一个 & 符号，将子进程放在后台执行。下列示例使用 **exec** 命令将原先收集的输出发送到打印机。这可以使脚本继续执行而不必等待 **print** 命令完成。

```
dcecp> exec lpr output.log &
dcecp>
```

第3章 编写脚本和 dcecp 对象

DCE 控制程序带有一些*对象*，提供对 DCE 单元中可管理组件进行管理访问。例如，**principal** 对象允许管理员管理安全性服务注册表数据库中的主体信息。同样，**rpcgroup** 对象允许管理员管理 CDS 中的组信息。

某些 DCE 操作影响多个组件，例如在向 DCE 单元中添加新的用户时必须执行多个操作。为满足这一要求，DCE 控制程序提供了*任务对象*，使得管理员使用单一操作就可以对多个组件进行操作。例如，**user** 任务对象执行多个操作，包括在注册表中创建主体信息，将主体添加到组织和适当的组，为用户创建一个 CDS 目录等等。任务对象的外观和表现和其它 **dcecp** 对象相似，也同样实现了其它 **dcecp** 对象所使用的帮助系统。不过任务对象是用 **dcecp** 语言编写的，而不是用 C 语言编写的。这使得管理员可以很容易地对现存的脚本进行扩展或定制。

尽管 DCE 控制程序提供了任务对象来处理某些多组件的操作，但是由于单元配置的不同以及管理员对单元管理方式的差异，由 DCE 提供的任务对象难以满足每个 DCE 单元的全部要求。例如，单元可能使用 DFS 组件，或者单元可能实现一个和标准 OSF DCE 实现不同的单元目录命名方案。或者，某些 DCE 实现可能带有一些专门的管理组件，例如服务或库，需要使用不同的 **dcecp** 对象来管理。

为了适应一个单元的特定要求，DCE 控制程序语言允许管理员创建他们自己的脚本。管理员也可以扩展或修改任务对象，或者创建新的任务对象来管理 DCE 单元中的专门组件。本章提供扩展、修改或创建下列类型 dcecp 脚本的信息:

- 非正式管理脚本
- 正式任务对象

非正式管理脚本

非正式脚本允许管理员将多个操作存储在一个文件中，并在任何必需的时候重新执行这些操作。非正式脚本非常适用于那些只使用一两个参数的操作或执行简单任务的操作。而且，可以根据作者的需要对脚本的准确行为和输出进行定制剪裁。虽然非正式脚本可以在单元中的管理员之间共享，但是它们通常只是包含在其作者的 **.dcecprc** 文件之中。

脚本通常由一个或多个用 **proc** 命令创建的过程所组成。这样可以在 **dcecp** 提示符下输入过程名来调用脚本中的操作。

下列简单脚本打印关于当前单元和登录身份的信息:

```
# Show your current login name and your current cell name.
proc _dcp_whoami {} {
  global _c _u
  puts stdout "You are '$_u' logged into '$_c'."
}
```

可以在 **.dcecprc** 文件中直接包含这一脚本，或者使用 **source** 命令来包含，从而将实际的脚本保存在一个外部文件中。第二个方法允许其它管理员包含这一相同的脚本，只要在他们的 **.dcecprc** 文件中简单地用 **source** 命令指向该脚本文件即可。这个方法可以使 **.dcecprc** 文件更清晰，便于别人理解。作为替代方法，可以将脚本放在 **init.dcecp** 文件中，或使用指针指向脚本。对这个文件的修改适用于主机中的所有用

户。如需有关 **init.dcecp** 文件和 **.dcecprc** 文件的详细信息，请参阅第3页的『第1章 DCE 控制程序介绍』中的第15页的『定制 dcecp 会话』部分。以下是 **.dcecprc** 文件中 **source** 命令的一个示例:

```
source /usr/users/wardr/dcecp/local_lib.dcp
```

.dcp 文件扩展名是一种约定，用来命名由 DCE 控制程序使用的文件。另一个约定是在过程名称前加 **_dcp**，例如 **_dcp_whoami**。许多 **dcecp** 过程采用这个约定，以便将它们的名称和那些不使用这个约定的、由用户创建的过程区分开来。如果发现像 **_dcp_whoami** 这样的过程名很难记或者难以输入，可以对其重新命名。例如，通过在 **.dcecprc** 文件中使用 **rename** 命令，可以将过程重命名为 **whoami**，如下:

```
rename _dcp_whoami whoami
```

重新启动 **dcecp** 使更改生效，现在就可以在 DCE 提示符下输入 **whoami**，如下:

```
dcecp> whoami
You are 'cell_admin' logged into '/.../my_cell.goodco.com'.
dcecp>
```

通过将多个操作链接在一起，可以创建完成更多工作的脚本。例如，下列脚本列出了在一个 DCE 单元中所有的主机，然后检查每个主机在 CDS 中是否有一个用于 dts 实体的对象项。(这表示在该主机上有一个 DTS 服务器。)对于每个带有用于 dts 实体对象项的主机，脚本执行一个 **clock show** 操作，返回该主机上的时间。这个脚本在显示器上打印这些信息，将信息重新格式化使其更便于阅读，并循环处理单元中所有的主机，直到检查完所有的主机项。

采用前面说明过的同样方法，可以使 **_dcp_show_clocks** 过程对您的 **dcecp** 会话可用。

```
# Show the time on all of the dts servers running in your cell.
proc _dcp_show_clocks {} {
    set x [directory list /.:/hosts]
    foreach n $x {
        if {[catch {object show $n/dts-entity}] == 0} {
            set index [string last "/" $n]
            set y [string range $n [incr index] end]
            if {[catch {clock show $n/dts-entity} msg] == 0} {
                set i [expr 20 - [string length $y]]
                puts [format "Time on $y is %${i}s %s" " " \
                    [clock show $n/dts-entity]]
            } else {
                set i [expr 20 - [string length $y]]
                puts [format "Time on $y is %${i}s %s" " " \
                   "Server not responding."]
            }
        }
    }
}
```

正式任务对象

某些 DCE 环境可能有一些标准 DCE 控制程序对象所难以满足的特殊管理需要。在编写并分发一些非正式脚本来满足这些管理要求的同时，您可能需要用某种方式来将它们的操作文档化。而且更重要的是: 一个复杂操作可能需要使用多个选项来准确地控制脚本的行为。您可以使用现有的帮助系统和 **parseargs** 功能，就像那些由 **dcecp** 提供的正式任务对象所使用的那样，而不是发明一种新的机制来提供帮助并处理复杂的参数分析操作。采用这种方式可以使您的脚本和其它 **dcecp** 一致。

正式任务对象建立在前面说明的非正式脚本的思想基础上，并带有一些重要的补充:

- 在脚本开始处的参数表将操作定义为脚本中不同的过程。参数表也可以定义可用的选项。调用一个 **parseargs** 过程来分析调用脚本时传递的参数和选项。
- 每个操作的帮助信息都放在脚本的参数表中。其它脚本用户可以使用标准的 **dcecp help** 操作来获取这些信息。
- 包含大量错误控制，因为很难预测或控制脚本执行时的具体情况。

本节的其余部分说明正式任务对象所采用的一些通用结构和约定。为了帮助说明，我们采用 DCE 控制程序所提供的 **dcecp user** 任务对象。

任务对象模型

这里分析 **user** 任务对象中那些您在用 DCE 控制程序编写其他任务对象时应当模仿的部分。遵循这一基本模型可以保证您任务对象的外观和行为与其它 **dcecp** 任务对象一致。

为提高效率并便于阅读，示例中没有包含 **user** 任务对象的所有过程。而且我们在代码示例中省略了所包含的过程中的某些重复部分，用垂直的省略号表示省略部分。完整的 **user** 任务对象存放在 *dcelocal*/**dcecp** 下。

根据所操作的实体来命名您的对象，而不应当用一个动词，例如 ″show″ 或 ″modify″ 来命名对象。DCE 控制程序对象是根据它们操作的 DCE 实体来命名的。如 **rpcentry** 和 **principal** 这样的原始对象对单一可管理的 DCE 实体进行操作。任务对象在更高的层次上进行操作，通常调用多个原始对象来完成其目标。user 任务对象的作者创造了作为可管理对象的高层实体——*user*。

user 对象以顶层的 **proc** 命令及其参数表开始，参数表定义 **user** 对象提供的过程和操作。使用下列语法在参数表中定义各个过程:

```
verb command function_call  procedure_name "helptext_string"
```

对 **parseargs** 过程（定义在一个叫做 **parseargs.dcp** 的独立文件中）的调用将返回要调用的内部过程的名称及其参数。**parseargs** 过程在第52页的『使用 parseargs 过程』中说明。

```
# proc user - This procedure is the front end for the user task
# scripts.  All argument checking for the provided switches is done
# in the individual functions.
#
proc user { args } {
  set arg_table {
    {create command function_call _dcp_create_user
       "Create a DCE user" }
    {delete command function_call _dcp_delete_user
       "Delete a DCE user"}
    {show   command function_call _dcp_show_user
       "Show the attributes of a DCE user"}
    {help   help  help_list
       "Print summary of command-line options and abort"}
    {operations operations operation_list
       "Return valid operations for command."}}
  set verbose_prose
"This object allows the manipulation of a DCE user.  A user is
represented as a principal and account with membership in a group and
organization as well as having a directory in the CDS namespace.  A user
may be created, deleted or have attribute information returned.  The
argument is a list of either relative or fully qualified principal names.
```

```
All fixed attributes of the principal and account object may be specified
when creating a user.  The -force option to the create verb allows the
group or organization for that user to be created if necessary.  The user
is provided a directory in the CDS namespace, with the appropriate ACLs.
Access to create a user requires the correct ACLs on principal, group and
organization directories within the registry and the clearinghouse and
users directory in the CDS namespace."
  set local_args $args
  parseargs $arg_table local_args -found_one

  if { [info local help_prose ] > 0 } { return $help_prose }
  if { [info local function_call ] > 0 } {
    return [$function_call local_args]
            } else {
    error "\"user\" object requires a verb to form a command."
  }
}
```

脚本的下一部分检查 **_dcp_create_user** 过程，它使用许多选项或属性作为输入。虽然这个过程依赖多个低层过程来完成创建用户的实际工作，不过在示例的开始处仅说明一个低层过程 **_dcp_create_principal_entry**。

然后脚本继续说明 **_dcp_create_user** 过程。注意，这个过程（和所有低层过程）的名称以下划线开始。这是因为经常要使用 Tcl **info** 命令来返回所有过程的名称，采用这个约定可以把这些内部过程和类似 **user** 的那些已经文档化的过程区分开来。而且名称的 **_dcp** 部分可以区分 **dcecp** 过程和主机中其它 Tcl 过程。

_dcp_create_user 过程有一个定义可用选项的参数表。这个参数表和脚本的初始参数表不同，它缺少 **command** 关键字和定义脚本中各个过程的 **function_call** 变量。

接着让初始化选项或列表中某个属性输入的变量。**process_attribute_list** 过程（在示例的末尾）实际对作为一个列表传递的属性进行语法分析。然后执行在注册表和 CDS 中创建用户信息的工作。接近末尾的清除过程 **_dcp_cleanup_user_create** 可以撤销一个失败的用户创建操作。

```
 .
 . [several low-level procedures omitted]
 .
#
# This procedure creates a principal in the current registry _s(sec)
# if that principal does not yet exist.
#
proc _dcp_create_principal_entry { principal_name princ_args} {
  set list_of_principals [principal catalog]
  if { [lsearch $list_of_principals $principal_name] == -1} {
    if { [llength $princ_args ] != 0 } {
      principal create $principal_name -attribute $princ_args
    } else
            } else {
    error "Principal \"$principal_name\" already exists."
  }
}
#
# proc _dcp_create_user - This procedure actually creates a DCE user.
# Several steps are performed.  If the principal does not exist
# a new one is created.  If the groups do not exist and a -force switch is
# set, then two new groups will be added.  The user will be added to the
# groups.  The account will then be created.  An entry in the CDS
# namespace will then be created with the appropriate ACLs.
#
proc _dcp_create_user { local_args } {
  set arg_table {
```

```
    {-alias string alias
          "Add principal named as an alias of specified uid."}
    {-attribute string attribute_list
          "Provide attributes in an attribute list format."}
    {-client string client
          "Can the account principal be a client."}
    {-description string descr
           "A general description of the account."}
    {-dupkey string dupkey
           "Can the accounts' principal have duplicate keys."}
    {-expdate string expdate
           "When does the account expire."}
 .
 . [repetitive elements omitted]
 .
    {-uid integer uid
         "User Identifier of the principal to be added."}}
#
# Initializing some variables.
#
upvar 1 local_args cargs
    set local_args $cargs
    set account_args ""
    set princ_args ""
    set group_args ""
    set force 0
    parseargs $arg_table local_args -no_leftovers
    if { [info local help_prose ] > 0 } { return }
    if { [llength $local_args] > 1 } {
      error "Unrecognized argument [lindex $local_args 1]."
    } elseif { [llength $local_args] == 0 } { error "No user name."
    } else { set account_name $local_args }
#
#  If parseargs returned attributes in a list instead of options,
#  create an attribute list.  Then call process_attribute_list to
#  parse the list.
#
    if { [info local attribute_list] > 0} {
       set pile_of_attributes "alias client descr dupkey expdate\
       forwadabletkt fullname force group home organization maxtktlife \
       maxtktrenew mypwd password postdatedtkt proxiabletkt pwdvalid \
       renewabletkt server quota shell stdgtauth"
       process_attribute_list attribute_list $pile_of_attributes
    }
#
# If user entered attributes as options rather than in a list,
# check for attribute options.
#
    if { [info local group] > 0} {
      set account_args [format "%s {%s %s}" $account_args group $group]
    } else { error "No group name specified." }
    if { [info local organization] > 0} {
      set account_args [format "%s {%s %s}" $account_args organiz \
  $organization]
    } else { error "No organization name specified." }
    if { [info local password] > 0} {
      set account_args [format "%s {%s %s}" $account_args password \
  $password]
    } else { error "No password specified." }
    if { [info local mypwd] > 0 } {
      set account_args [format "%s {%s %s}" $account_args mypwd $mypwd]
    } else { error "No admin password specified." }
#
# principal and group operations both use the principal's fullname
#
    if { [info local fullname] > 0 } {
      set princ_args [format "%s {%s {%s}}" $princ_args fullname \
```

```
  $fullname]
      set group_args [format "%s {%s {%s}}" $group_args fullname \
  $fullname]
    }

    if { [info local uid] > 0 } {
      set princ_args [format "%s {%s %s}" $princ_args uid $uid]
    }
 .
 . [repetitive elements omitted]
 .

    if { [info local stdtgtauth] > 0 } {
       set account_args [format "%s {%s %s}" $account_args stdtgtauth \
           $stdtgtauth]
    }
#
# set variables if entered as attributes in an attribute list
#
    set account_name [lindex $account_name 0]
    set group_created 0
    set org_created 0
    set group_arg ""
    set org_arg ""
#
# do the work - create principal, do group and organization
# operations, create the account, and create directory in CDS
#
    foreach element $account_name {
      set clup_user "_dcp_cleanup_user_create $element -principal"
      _dcp_create_principal_entry $element $princ_args

      if { $force == 1 } {
   if {[ catch {_dcp_create_group $group group_created} \
    msg] != 0 } {
              _dcp_cleanup_user_create $element -principal
      error $msg
          }
          if { $group_created == 1 } {
             set group_arg "-group group"
          }
          if {[ catch {_dcp_create_org $organization org_created} \
    msg] != 0 } {
              set clup_user [concat $clup_user $group_arg]
              eval $clup_user
      error $msg
          }
          if { $org_created == 1 } {
              set org_arg "-org organization"
          }
      }
      set clup_user [concat $clup_user $group_arg $org_arg]
      if {[catch {_dcp_add_group_entry $group  $element} msg] != 0} {
          eval $clup_user
      error $msg
      }

      if {[catch {_dcp_add_org_entry $organization $element} msg] != 0 }
{
  eval $clup_user
      error $msg
      }

      if {[catch {_dcp_add_account_entry $element $account_args} \
msg] != 0} {
          eval $clup_user
      error $msg
```

```
      }

      if {[catch {_dcp_add_namespace_entry $element} msg] != 0} {
          eval $clup_user
      error $msg
      }
    }
    set _n $account_name
    return
}
#
# _dcp_cleanup_user_create  - This function undoes changes after a
# failure in one of the user create functions as though the operation
# never occurred
#
proc _dcp_cleanup_user_create {account_name args} {

     if { [lsearch $args -principal] != -1 } {
        principal delete $account_name
     }
     if { [lsearch $args -group] != -1 } {
        upvar 1 group clean_group
        group delete $clean_group
     }
     if { [lsearch $args -org] != -1 } {
        upvar 1 organization clean_org
        organization delete $clean_org
     }
}
#
# process_attribute_list - Takes an attribute_list and parses out the
#                          appropriate attributes contained in the
#                          pile_of_attributes variable
#
proc process_attribute_list {attribute_list pile_of_attributes} {
   foreach element $pile_of_attributes { upvar 1 $element _dcp_$element
}
   upvar 1 attribute_list _dcp_attribute_list
   set _dcp_attribute_list [check_list_list $_dcp_attribute_list]
   foreach element $_dcp_attribute_list {
       if { [llength $element] != 2 } {
           error "Incorrect attribute list element
       }
       set attribute_name [lindex $element 0]
       set attribute_value [lindex $element 1]
       set _dcp_attr_name [info vars _dcp_$attribute_name*]
       if {[llength $_dcp_attr_name] > 1} {
           error
            "Ambiguous attribute "\$attribute_name\" could be:
$_dcp_attr_name."
       }
       set [set _dcp_attr_name] $attribute_value
   }
}
proc check_list_list {attribute_list} {
   set not_list_list 0
   set i 1
   foreach element $attribute_list {
        if {[llength $element] != 2 && [llength $attribute_list]
< 3} {
           if {$i == 1} {
              return [format "{%s}" $attribute_list]
           }
        }
```

```
        incr i
    }
    return $attribute_list
}
```

我们在 **user** 任务对象中讨论的下一个过程是 **_dcp_show_user**，它使用一个可选的参数，并且返回许多输出信息。这个过程返回 **principal show** 和 **account show** 操作的结果。

```
#
#_dcp_show_user - This procedure shows the principal and account
#                 attribute lists for a specified user.
#
proc _dcp_show_user {local_args} {
upvar 1 local_args cargs
    set local_args $cargs

    parseargs "" local_args -no_leftovers
    if { [info local help_prose ] > 0 } { return }
    if { [llength $local_args] > 1 } {
      error "Unrecognized argument [lindex $local_args 1]."
    } elseif { [llength $local_args] == 0 } { error "No user name."
    } else { set account_name $local_args }
    # Take the first element of the account_name in order to
    # eliminate list nesting.
    set account_name [lindex $account_name 0]
    set _dcp_principals [principal catalog -simplename]
    # Show each account that has been requested.
    foreach element $account_name {
      if { [lsearch $_dcp_principals $element] == -1 } {
error "User \"$element\" does not exist."
            } else {
set _dcp_user_attributes [principal show $element]
      }
      set _dcp_accounts [account catalog -simplename]
      if { [lsearch $_dcp_accounts $element] == -1 } {
error "User \"$element\" does not exist."
            } else {
set _dcp_user_attributes [format "%s\n%s" \
    $_dcp_user_attributes [account show $element -all]]
      }
    }
    return $_dcp_user_attributes
}
```

使用 parseargs 过程

使用参数或选项的任务对象可以调用 **parseargs** 过程来对调用脚本或对象时所传递的参数进行语法分析。**parseargs** 过程是一个在单独文件中的脚本，这为在 **dcecp** 脚本中进行参数语法分析提供了方便而且可以重用的方法。基本语法为:

parseargs *parse_options local_args args*

这个过程依赖于由调用它的脚本所传递的参数。**parseargs** 过程需要下列输入:

parse_options

描述分析选项的参数表 (**arg_table**)。*parse_options* 参数可以由五个元素组成，例如在脚本的顶级参数表中，也可以由四个元素组成，例如脚本中被调用的过程的低层参数表中。*parse_options* 的两个语法为:

verb **command** *variable command_name* "*help string*"

或者

-options type variable "help string"

verb 提供顶级分析。通常一个操作包含一个对象和一个动词。动词部分通常调用另一个过程。

command
一个关键字，表示被定义的过程是对象的一个动词。

variable
存放选项值的变量的名称。在分析动词时，变量被命名为 **function call**。在分析选项时，变量被命名为正在语法分析的选项。例如，如果选项的名称是 **-alias**，变量被命名为 **alias**。

command_name
要在变量中存储的过程的名称。

help string
描述动词或选项用法的字符串。

-options
要被语法分析的选项的实际字符串值，例如 **-attribute** 或 **-mypwd**。

type 和 **-option** 关联的变量的类型。可以接受的类型为 **integer**，**string**，**float**，**boolean**，**command** 和 **help**。

local_args
将要被语法分析的参数。**parseargs** 过程将所有已识别的项抽取到一个列表中，然后将 *local_args* 重新设置成没有语法分析（或不可语法分析）的值。例如，像 **user create** 这样的顶级命令包含一些选项，而这些选项只有在过程实现 **create** 操作时才进行分析。

args 一两个标志之一：

-found_one
通知语法分析器在找到一个过程参数之后返回。例如在 **user create** 中，语法分析器应当在找到并处理一个 **create** 命令后返回。

-no_leftovers
查找额外选项，如果找到则生成一个错误。

调用任务对象

在编写好任务对象之后，必须使其可供使用。如果脚本只供自己个人使用，可以在自己的 **.dcecprc** 文件中包含它，并如第45页的『非正式管理脚本』中描述的那样调用它。.

正式任务对象则需要采取一些步骤使它们的行为和其它 **dcecp** 对象相似。

1. 以 **root** 登录并将完成的脚本复制到 *dcelocal***/dcecp** 目录下，并且将文件许可权设置为可执行。
2. 启动 **dcecp** 并运行 **auto_mkindex** 实用程序。这将创建通知 DCE 控制程序所有可用对象的信息。使用 root 特权在任务对象驻留的目录下运行下列命令，在 UNIX 系统中，这一目录通常是 *dcelocal***/dcecp**。

```
% dcecp
dcecp> auto_mkindex /opt/dcelocal/dcecp *.dcp
dcecp>
```

3. 为了在 **dcecp** 帮助屏中包含新任务对象的名称，编辑文件 **/opt/dcelocal/dcecp/help.dcp**。这一文件在响应 **dcecp help** 操作时显示。

如果需要在其它 DCE 主机上执行该脚本，则应当使文件在该主机上可用，通常情况下，这意味着将该文件复制到该主机的 **/opt/dcelocal/dcecp** 目录下，然后对目录中的文件运行 **auto_mkindex** 实用程序。也可能需要将对象的名称放到 **/opt/dcelocal/dcecp/help.dcp** 文件中。

作为一种便捷的方法，可以用 DCE 控制程序的 **hostdata** 对象来编写一个脚本，在每个主机上创建文件。然后这个脚本可以使用 **hostdata** 对象的后处理器属性来运行 **auto_mkindex** 实用程序。第83页的『第9章 管理 DCE 主机服务和主机数据』 中包含使用 **dcecp hostdata** 对象的信息。

第2部分 DCE 管理任务

第4章 DCE 管理任务对象

*IBM DCE 版本 3.2 AIX 和 Solaris 版: 管理指南--核心组件*的这一部分讨论 DCE 所提供的管理任务对象的目的和用途。简而言之，这些特殊的 **dcecp** 对象通过将几个低层操作组合在一起来执行日常的高层管理任务。

通常单一任务对象使用或影响多个服务。例如，一个任务对象，**host** 对象可以将一个主机配置到一个 DCE 单元中。这一任务向 DCE 安全性服务、单元目录服务和主机守护进程服务添加特定类型的信息。因为 **host** 对象的单一调用可以执行多个步骤，它使管理员可以脱离低层的管理细节，否则就必须使用多个低层的 **dcecp** 管理对象才能完成。

在从高层讨论任务对象的同时，请记住实际有很多操作发生，只是我们没有详细描述而已。这些情况下，我们将指出在本指南的何处可以找到更详细的信息。这些通常指向本指南适当组件中相应的低层讨论部分。

利用任务对象简化 DCE 管理

单独的 DCE 控制程序对象对 DCE 中非常特定的信息进行操作。例如，**group** 对象完全针对 DCE 安全性服务注册表数据库的安全性组进行操作。**group** 对象允许管理员创建和删除安全性组，向安全性组中添加和除去成员，重命名组等。这种严格控制是必需的，因为这样允许对 DCE 进行定制裁剪，以满足非常特定的要求或环境。

虽然在配置一个新单元或修正某些访问控制问题时这些控制必需的，但是这样的控制会淹没日常的管理任务。作为例子，让我们看看向一个 DCE 单元添加新用户时所必需的最少步骤:

1. 使用 **principal** 对象为用户创建一个主体名称。
2. 使用 **group** 对象将主体添加到一个安全性组。
3. 使用 **organization** 对象将主体添加到一个安全性组织。
4. 使用 **account** 对象为主体创建一个帐户。
5. 使用 **directory** 对象为主体在 CDS 中建立一个目录。
6. 使用 **acl** 对象为主体提供访问 CDS 目录的权限。

在一个 15 到 20 用户的小单元中执行这六个步骤是不会有什么问题的。但是考虑一下一个带有成百上千甚至更多用户的单元的情况，以及将各种管理任务自动化的需要。

为了满足这种管理要求，DCE 控制程序包含多个管理*任务对象*，用于执行日常的 DCE 管理任务。这里，我们使用术语*任务*表示需要进行多个步骤操作的事情，例如添加一个用户由六个低层操作组成。

user 对象是一个任务对象，用于在 DCE 环境中添加并删除用户。例如，对 **user** 对象的单一调用可以执行前面提到的所有六个步骤，以便向 DCE 环境中正确地添加一个新的用户。也可以用这个任务对象从环境中删除用户。

通过使用 DCE 控制程序语言将这些任务对象作为 **dcecp** 脚本实现，这意味着可以根据需要扩展或更改它们的行为。例如， **user** 任务对象的缺省实现不对 DFS 信息进行

操作。如果您的 DCE 环境包含这个扩展服务，则可能需要向脚本中添加某些 DFS 操作。第1页的『第1部分 DCE 控制程序』 部分说明如何使用 DCE 控制程序语言编写及修改一个 **dcecp** 任务对象。

超越工具

虽然您使用任务对象来执行各种管理操作，但是您的主要重点是正在管理的元素或实体。DCE 提供的四个任务对象中的每一个都使您能够管理 DCE 单元中的某个特定元素或实体。这些元素如下:

DCE 单元

使用 **cell** 任务对象，可以测试一个单元是否正在运行，显示单元中可用服务的常规信息，或者备份安全性和 CDS 信息等等。

单元名称

可以创建并管理单元别名，用于在多个全局的目录服务中注册单元。这些操作使用 **cellalias** 任务对象。

DCE 主机

使用 **host** 任务对象，可以配置和删除一个单元中的 DCE 主机，显示主机的信息，或启动和停止主机上的 DCE 进程。

DCE 用户

可以使用 **user** 任务对象添加和删除一个 DCE 单元中的用户，或者显示用户的信息。

本部分的其余章节讨论如何用 DCE 提供的四个 **dcecp** 任务对象的缺省实现来管理这些 DCE 元素。

第5章 管理 DCE 单元

从单元管理员的角度来看，DCE 单元由一组网络服务组成，这些网络服务支持分布式应用程序的执行。然而这个简单语句并没有说明单元中任何可用的服务。实际上，DCE 服务器的数量和位置在每个单元中都是不同的。即使在同一个单元中，主机和网络的停机和重新配置都会影响服务的可用性。

虽然可以使用和各种服务相关的 **dcecp** 对象来测试单元中的服务是否可用以及这些服务在哪里等等，但是这是相当麻烦的。实际上 DCE 控制程序提供了一个 **cell** 任务对象，可以方便地列出已配置的服务，并测试它们是否可用。它同样可以备份由 DCE 安全性服务和 CDS 维护的关键数据。

显示所有已配置的 DCE 服务器和 DCE 主机

有些 DCE 单元可能会相对稳定，只有少数 DCE 主机或 DCE 服务器被添加进来或者从中除去；而其它的单元则可能相当活跃，每周甚至每天都要添加、除去或移动 DCE 主机或服务器。在这种环境中，跟踪 DCE 资源的位置变得非常困难，所以 **cell** 任务对象有一个 **show** 操作，查找单元中的各种数据库并返回已配置的 DCE 服务器和 DCE 主机的名称。

cell show 命令的一个用途是跟踪性能问题。例如，可能添加了很多的主机和用户，但是 CDS 或安全性服务器的数量或位置并没有相应地增加。或者也许您刚被雇来管理一个新的单元，您希望了解单元的构成情况。

输入 **cell show** 操作，可以显示一个单元中配置的 DCE 服务器和主机。这个命令返回根据类型分组的服务器列表和 DCE 主机的列表，如下:

secservers
: 每个值是一个安全性服务器的名称。

cdsservers
: 每个值是一个运行 CDS 服务器的机器的名称。这个名称是 **/.:/hosts** 下的简名。在该机器中必须配置一个中央交换库。

dtsservers
: 每个值是单元中的一个 DTS 服务器的名称。

hosts
: 每个值是单元中一个主机的名称，包含上述作为服务器的机器。这是 **directory list /.:/hosts** 操作的返回值。

下列示例显示本地单元中所有已配置的 DCE 服务器和主机的名称:

```
dcecp> cell show
{secservers
 /.../my_cell.goodco.com/subsys/dce/sec/earth}
{cdsservers
 /.../my_cell.goodco.com/hosts/earth}
{dtsservers
 /.../my_cell.goodco.com/hosts/krypton}
{hosts
 /.../my_cell.goodco.com/hosts/earth
 /.../my_cell.goodco.com/hosts/jupiter
 /.../my_cell.goodco.com/hosts/krypton
 /.../my_cell.goodco.com/hosts/mars
 /.../my_cell.goodco.com/hosts/mercury
```

```
 /.../my_cell.goodco.com/hosts/neptune
 /.../my_cell.goodco.com/hosts/pluto
 /.../my_cell.goodco.com/hosts/saturn
 /.../my_cell.goodco.com/hosts/uranus
 /.../my_cell.goodco.com/hosts/venus}
dcecp>
```

如果您有足够的许可权，只要把另一个单元的名称作为参数，就可以显示该单元中配置的 DCE 服务器和主机，如下列示例:

```
dcecp> cell show
/.../their_cell.goodco.com
{secservers
 /.../their_cell.goodco.com/subsys/dce/sec/gold}
{cdsserver
 /.../their_cell.goodco.com/gold}
{dtsservers
 /.../their_cell.goodco.com/hosts/silver/dts-entity}
{hosts
 /.../their_cell.goodco.com/hosts/brass
 /.../their_cell.goodco.com/hosts/bronze
 /.../their_cell.goodco.com/hosts/copper
 /.../their_cell.goodco.com/hosts/gold
 /.../their_cell.goodco.com/hosts/iron
 /.../their_cell.goodco.com/hosts/mercury
 /.../their_cell.goodco.com/hosts/silver
 /.../their_cell.goodco.com/hosts/steel
 /.../their_cell.goodco.com/hosts/tin}
dcecp>
```

测试单元操作

在发生客户机 / 服务器通信问题时，很容易就会推断出可能是单元中一个或多个 DCE 服务有问题。使用 **cell ping** 操作可以很容易地测试出一个单元的 DCE 服务是否正在运行。

如果调用时不使用选项，**cell ping** 操作对单元中的主安全性服务器、带有主中央交换库的 CDS 服务器和所有的 DTS 服务器执行一个 **server ping** 操作。使用 **-replicas** 选项可以和主服务器一起测试 CDS 和安全性服务的副本。**-clients** 选项测试单元的每个 DCE 主机，即循环查看 CDS 的 **/.:/hosts** 目录，使用主机名作为参数对每个主机执行 **host ping**。

如果出现故障，这一操作生成一个错误，并返回无法联系的服务器或主机的列表。如果成功，操作返回消息 DCE Services Available。如果使用 **-clients** 选项时都成功，返回的消息是 DCE Clients Available。

下列示例 ping 本地单元中配置的所有主 DCE 服务器。

```
dcecp> cell ping
DCE services available
dcecp>
```

下列示例 ping 本地单元中配置的所有 DCE 主机。这个命令可能需要很长时间（几分钟或更多）才能完成，具体取决于单元的大小和设置的超时值。

```
dcecp> cell ping -clients
DCE clients available
dcecp>
```

如果您有足够的许可权，只要把另一个单元的名称作为参数，就可以 ping 该单元中配置的 DCE 服务器和主机，如下列示例:

```
dcecp> cell ping /.../their_cell.goodco.com
DCE services available
dcecp>
```

备份安全性服务注册表和 CDS

注: 下列讨论仅针对旧的 DCE 。如需有关 DCE 安全性注册表和 LDAP 集成功能的信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

由于组织在日常操作中越来越依赖于 DCE 单元，他们无法承担失去单元目录和安全性数据的损失。组织通常使用备份系统来避免丢失上述及其它关键数据。但是如果没有妥善地保护档案的内容，用传统备份方法备份这些 DCE 数据库会引发安全性漏洞。

幸运的是 DCE 包含将这些必不可少的数据库备份到指定目的地的功能。一旦开始使用 DCE 的机制来备份 CDS 和安全性数据，就可以重定向传统的备份程序以忽略这些 DCE 数据库。

cell backup 操作备份单元中的主安全性数据库和每个主副本的中央交换库。这个操作的前提条件是在每个需要备份的主机上必须都运行着一个 **dced** 程序。

为给单元的日常备份做准备，设置一个指定备份目的地（通常是一个磁带档案）的扩展注册表属性 (ERA)，然后将属性添加给这些主体: 主安全性服务注册表数据库和所有需要备份的主副本的 CDS 中央交换库。参照这些步骤:

1. 作为单元管理员，创建一个字符串的 ERA，指定备份的目的地。将 ERA 命名为 **/.:/sec/xattrschema/bckp_dest**，类型为 **printstring**。选择名称为 **principal** 的 ACL 管理器，并将其许可权位设置为 **r**（**读**）、 **m**（**管理**）、**r**（**读**）和 **D**（**删除**），如下:

   ```
   dcecp> xattrschema create /.:/sec/xattrschema/bckp_dest \
   > -encoding printstring -aclmgr {principal r m r D}
   dcecp>
   ```

2. 作为单元管理员，将新的 ERA (**bckp_dest**) 添加到主体 **dce-rgy**（DCE 安全性服务注册表数据库）。将值设置为 **tar** 文件名或作为备份目的地的设备:

   ```
   dcecp> principal modify dce-rgy -add {bckp_dest tarfilename_or_device}
   dcecp>
   ```

3. 作为单元管理员，将新的 ERA (**bckp_dest**) 添加到主体 **/.:/hosts/***hostname***/cds-server**（CDS 服务器）。将值设置为 **tar** 文件名或作为备份目的地的设备:

   ```
   dcecp> principal modify /.:/hosts/hostname/cds-server \
   > -add {bckp_dest tarfilename_or_device}
   dcecp>
   ```

通过备份或恢复 CDS 服务器中的下列目录来备份或恢复 CDS 服务器:

```
/opt/dce/local
/var/dce
/krb5
/etc/dce
```

然后无论在何时都可以调用 **cell backup** 操作来备份注册表数据库或 CDS 数据库，如下:

```
dcecp> cell backup
dcecp>
```

通过将另一个单元的名称指定为 **cell backup** 的操作参数，可以备份该单元。注意您必须有该远程单元相应的许可权。（有关必要的特权，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference*。）

更改 DCE 服务器的 IP 地址

服务器更新

使用下列过程修改被配置为 CDS 和安全性服务器的机器的 IP 地址。

1. 如果系统中正在运行一个 DFS 服务器或任何 DCE 应用程序，则应当先停止它们，并且禁用 **auto-restart**，对于 AIX 是在 **inittab** 中，对于 Solaris 是在 **/etc/rc*.d** 中。
2. 删除这个机器上中央交换库的内容，它将在 IP 地址更改后被复原。如果不知道名称，使用 **cdscp show cell** /.: 命令可以找到。

   ```
   cdscp clear clearinghouse /.:/<host_ch>
   ```

3. 停止机器上的所有 DCE 守护进程:

   ```
   stop.dce
   ```

4. 用一个命令除去端点数据库、clerk 高速缓存和老的凭证:

   ```
   clean_up.dce
   ```

5. 编辑文件 **/opt/dcelocal/etc/security/pe_site** 反映新的地址，以便启动安全性。
6. 如果 **opt/dcelocal/var/dced/cdscache.inf** 存在，修改其中的 IP 地址以便 **cdsadv** 可以查找到目录服务器。
7. 更改系统的 IP 地址并重新引导。如果 DCE 没有被 inittab 自动启动，使用 **start.dce** 来启动。**gdad** 和 **dtsd** 守护进程不会起来，因为 CDS 还不能完全地工作。在转换进程完成后再启动这些守护进程。当系统检查 **dced** 注册（显示一个等待消息）时按 <**Ctrl-C**> 退出。
8. 因为 CDS 不可用，设置 **BIND_PE_SITE** 环境变量。

   ```
   export BIND_PE_SITE=1
   dce_login cell_admin
   ```

9. 标识 CDS 要管理的中央交换库（确保其名称和前面使用 **clear clearinghouse** 命令时所指定的完全相同）:

   ```
   cdscp create clearinghouse /.:/<host_ch>
   ```

10. 因为 CDS 在启动时不知道这个中央交换库，所以 **cdsadv** 进程也同样不知道。编辑文件 **/opt/dcelocal/etc/security/pe_site** 反映新的地址，以便启动安全性服务。重建职员程序高速缓存:

    ```
    stop.dce cds_srv cds_cl cds_second
    cd /opt/dcelocal/var/adm/directory/cds
    rm cds_cache.* cdsclerk_*
    start.dce cds_srv cds_cl cds_second
    ```

    当系统检查 dced 注册（显示一个等待消息）时按 <**Ctrl-C**> 退出。

11. 现在这一 CDS 和安全性服务器被重新配置为使用新的 IP 地址了。取消 **BIND_PE_SITE** 环境变量的设置，并验证可以像往常一样使用 **dce_login**:

```
unset BIND_PE_SITE
dce_login cell_admin
```

12. 验证可以成功地访问名称空间:

```
    cdsli -o
/.:/cell-profile
/.:/fs
/.:/lan-profile
/.:/sec
```

13. 在 CDS 中更新服务器的 self 项。如果系统中运行 DFS 服务器或其它依赖 self 项的应用程序，这一步骤就非常关键。

```
rpccp unexport -i elaf8308-5dlf-11c9-91a4-08002b14a0fa,3.0 \
               /.:/hosts/<server_name>/self
rpccp export -i elaf8308-5dlf-11c9-91a4-08002b14a0fa,3.0 \
               -b ncadg_ip_udp:<new_ip_addr>[135] \
               /.:/hosts/<server_name>/self
```

14. 运行 **stop.dce**.
15. 运行 **start.dce**.

客户机更新

在更改了一个 CDS 或安全性服务器的 IP 地址后，需要在客户机中做下列更改。如果系统中运行 DFS 或任何 DCE 应用程序，则需要先停止它们。

1. 停止机器上所有的 DCE 守护进程:

```
stop.dce
```

2. 删除端点数据库、职员程序高速缓存（它引用 CDS 服务器的旧 IP 地址）和安全性凭证。

```
clean_up.dce
```

3. 更改文件 **/opt/dcelocal/etc/security/pe_site** 以便 **dced** 在重新启动时可以找到安全性服务器。
4. 更改 **/opt/dcelocal/var/dced/cdscache.inf** 文件中的 IP 地址，以便 **cdsadv** 可以查找目录服务器。如果文件 **/opt/dcelocal/var/dced/cdscache.inf** 不存在就创建它，格式为:

```
<cds_server_ip_addr> ip
```

   例如

```
9.3.53.233 ip
```

5. 启动 DCE 守护进程:

```
start.dce
```

6. 如果文件 **/opt/dcelocal/var/dced/cdscache.inf** 不存在则需要执行下列步骤。
   a. 设置 **BIND_PE_SITE** 变量，因为 CDS 访问还没有恢复。

```
export BIND_PE_SITE=1
dce_login cell_admin
```

   b. 通知 **cdsadv** 进程为 CDS 服务器使用新的 IP 地址:

```
cdscp define cached server <server_name> tower \
        ncadg_ip_udp:<new_address>
```

c. 现在这个客户机已经完全了解服务器的新 IP 地址了。请验证可以正常地 **dce_login**:

```
unset BIND_PE_SITE
dce_login cell_admin
```

7. 请验证可以成功地访问名称空间:

```
     cdsli -o
/.:/cell-profile
/.:/fs
/.:/lan-profile
/.:/sec
```

更改 DCE 客户机的 IP 地址

1. 启动 DCE 守护进程:

```
start.dce
```

2. 在 CDS 中更新客户机的 self 项:

```
rpccp unexport -i e1af8308-5d1f-11c9-91a4-08002b14a0fa,3.0 \
               /.:/hosts/<client_name>/self
rpccp export -1 e1af8308-5d1f-11c9-91af-08002b14a0fa,3.0 \
                -b ncadg_ip_udp:<new_ip_addr>[135] \
                 /.:/hosts/<client_name>/self
```

修改或扩展单元对象

cell 任务对象是作为脚本实现的，因此管理员可以根据现场的情况对其进行修改或扩展。这里是一些可能的修改或扩展:

- 添加一种显示 DFS 服务器信息的方法。
- 为 **cell show** 操作添加选项，以省略列出单元中所有的主机，或者只显示特定的 DCE 服务器。

本指南第1页的『第1部分 DCE 控制程序』讨论了用 **dcecp** 语言创建新 **dcecp** 对象或修改现存对象的方法。

第6章 管理 DCE 主机

较大的 DCE 单元可能包含许多主机，有些主机运行 DCE 服务器和应用程序服务器，还有些则作为客户机系统使用。有的还运行应用程序服务器，同时对驻留的用户也充当客户机。DCE 主机配置的这种灵活性使得单元中主机难以控制，而且很难跟踪主机中运行的服务器及其可用性。**host** 任务对象显示和主机关联的 DCE 进程和应用程序进程，使得管理员更容易地管理机器中的 DCE 服务器和应用程序进程。

使用 **host** 任务对象，可以显示单元中本地和远程主机的进程信息，可以在单元范围内的主机上启动和停止 DCE 进程，也可以在单元中配置本地 DCE 主机，或者从单元中除去（取消配置）远程 DCE 主机。在 **dcecp** 中使用 **host help** 和 **host operations** 命令可以获得这一对象的联机帮助。

所有对远程主机执行的 **host** 对象操作，除了 **host catalog** 之外，都需要在远程主机上运行 **dced**。

列出单元中的 DCE 主机

使用 **host catalog** 操作可以确定 DCE 单元中配置的 DCE 主机的数目和名称。这个操作可以帮助您确定某个特定的主机是否已经被配置到单元中。该主机是否运行和这个操作并没有关系，因为 **host catalog** 操作实际只执行一个 **directory list /.:/hosts** 操作，并不和该主机进行交互。这个方法依赖于每个主机都在 **/.:/hosts** 目录下注册其名称的约定。如果您的主机在其它目录下注册，则需要修改 **host** 任务对象中的 **host catalog** 操作。可以在第169页的『第18章 管理 CDS 目录』中读到 CDS 目录作用及其用途的详细信息。

host catalog 操作对 **cell show** 操作进行重新编排，不过并不单独显示 DCE 服务器。以下示例操作显示单元中配置的所有 DCE 主机:

```
dcecp> host catalog
/.../my_cell.goodco.com/hosts/bigbox
/.../my_cell.goodco.com/hosts/drifter
/.../my_cell.goodco.com/hosts/duh
/.../my_cell.goodco.com/hosts/heater
/.../my_cell.goodco.com/hosts/pc1
/.../my_cell.goodco.com/hosts/pc2
/.../my_cell.goodco.com/hosts/pc3
/.../my_cell.goodco.com/hosts/peewee
/.../my_cell.goodco.com/hosts/xoltar
/.../my_cell.goodco.com/hosts/xray
/.../my_cell.goodco.com/hosts/zoof
dcecp>
```

使用 **-simplename** 选项可以省略单元名称，如下列示例:

```
dcecp> host catalog -simplename
hosts/bigbox
hosts/drifter
hosts/duh
hosts/heater
hosts/pc1
hosts/pc2
hosts/pc3
hosts/peewee
```

```
hosts/xoltar
hosts/xray
hosts/zoof
dcecp>
```

显示 DCE 主机中配置的所有服务器

在较大的单元中，DCE 服务器和应用程序服务器往往驻留在多个主机上，可能经常需要查看配置在特定主机上运行的服务器。DCE 控制程序的 **host show** 操作读取一个 DCE 主机的服务器配置和执行信息，并返回一个配置在该主机上的服务器列表。这个列表包含每个服务器的简名，并指出每个服务器是否正在运行。这个列表同时还表示一个安全性服务器是主控还是副本，以及一个 DTS 实体是职员程序还是服务器。

这个操作依赖 **server** 对象（以及 DCE 主机守候进程）来显示已配置的服务器。可以在第93页的『第10章 DCE 应用程序管理』中读到控制服务器的更多内容。

下列示例显示配置成在 DCE 主机 **xoltar** 上运行的服务器:

```
dcecp> host show /.:/hosts/xoltar
video_clip running
dts-entity running clerk
dcecp>
```

注: 只有向 DCED 注册过的服务器才会被显示。向 DCED 注册的 DCE 服务器为: **dtsd**、**emsd**、**pwd_strengthd**、**gdad** 和 **auditd**。

测试 DCE 主机是否正在运行

因为 DCE 通信在客户机和服务器通信之前经常涉及多个步骤，所以往往很难诊断通信故障。例如，主机已经被配置到单元中，但是并没有在上面运行服务器，或者目前没有运行 DCE 服务。使用 **server ping** 操作可以测试一个服务器进程是否正在运行，如果失败，则可能需要一种方法来检查是否可以通过网络访问该 DCE 主机。DCE 控制程序的 **host ping** 操作测试一个主机的 DCE 服务在网络中是否可访问，返回 **1** 表示可以，**0** 表示不可以。

host ping 操作测试远程主机的 DCE 守护进程 (**dced**) 是否存在。可以在第83页的『第9章 管理 DCE 主机服务和主机数据』中读到 **dced** 的目的以及用途。

下列示例测试是否可以从网络中访问主机 **duh** 上的 **dced**:

```
dcecp> host ping /.:/hosts/duh
1
dcecp>
```

修改或扩展主机对象

host 任务对象是以脚本实现的，因此管理员可以根据现场的情况对其进行修改或扩展。例如，管理员可能需要在对象中添加 DFS 信息。也可以添加对专门命令的调用以启动或停止应用程序服务器。例如 **printer stop** 操作就可能很有用。

本指南第1页的『第1部分 DCE 控制程序』 讨论了用 **dcecp** 语言创建新 **dcecp** 对象或修改现存对象的方法。

第7章 管理 DCE 用户

一个使用最频繁的 DCE 管理任务可能就是管理 DCE 环境中的用户了。企业的重组、不断变化的商业需要以及经济动荡都带来种种压力，迫使人员进入或离开公司，或者在各种组和组织间流动。

DCE 用户代表设计 DCE 所要支持的一个部分，DCE 服务对用户进行认证，允许他们访问一些授权的服务，并拒绝对未授权服务的访问。实际上，用户有非常复杂的管理要求，他们的信息分布在多个帮助验证并控制其活动的服务之间。用户信息包含主体名称、组和组织信息、帐户信息和 CDS 中的信息等。

DCE 控制程序包含各种分离的管理对象，以管理 DCE 单元中用户信息的各个部分。虽然这些分离的对象对特定用户信息做些小的调整是非常有用的，但是它们在重复性任务中的使用则是令人生厌的，例如在单元中添加或删除用户。一个简单的方法是使用 **user** 任务对象在单元中更容易地创建、删除或显示用户信息。可以在 **dcecp** 中使用 **user help** 和 **user operations** 命令获得这个对象的联机帮助。

创建新用户

DCE 环境中的每个**用户**是一个带有唯一身份（主体名称）的人员。每个主体是至少一个安全性组和组织的成员，并且在安全性服务注册表数据库中有一个帐户。虽然不是必需的，每个主体在 CDS 中都可以有一个目录。

在使用 **user** 任务对象创建用户时，执行多个低层操作:

- **user create** 操作创建一个新的主体名称，并将这一主体名称添加到安全性组和组织中。如果在调用操作时这些安全性组或组织并不存在，可以使用 **-force** 选项来强制创建。主体的属性取缺省值，如果有必要也可以指定。*IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference*中列出了所有属性。

  通常，安全性组的名称包含在访问控制表 (ACL) 中，访问控制表是用来管理用户对 DCE 环境中各种服务器和数据对象访问权限的。一个安全性组织维护适用于该组织中所有主体的策略。策略控制如帐户寿命，帐户口令是否失效或何时失效，以及口令中是否可以包含非字母字符等等之类的内容。可以在第283页的『第30章 创建和维护主体、组和组织』中读到更多关于管理主体、组和组织的内容。
- **user create** 操作为主体创建一个帐户，并创建用户的口令。帐户的属性取缺省值，如果必要，也可以指定。*IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference* 中列出了所有属性。

  主体的帐户包含主体的这些信息: 组和组织的名称，帐户的创建和失效信息，以及票卷信息（对 DCE 环境中的资源标识该主体）。在第 31 章中可以读到关于管理帐户的更多内容。
- 最后，**user create** 操作对 CDS 添加一个 **/.:/users/***principalname* 的目录。这个目录可以存储用户特定的应用程序位置信息。这一操作也向缺省的 ACL 中添加一个 ACL 条目，赋予用户对目录的 **rwtci** 许可权。这些许可权允许用户插入并链接对象，但是不能删除该目录或管理目录的副本。而且用户不能创建其它的目录，除非赋予他们对中央交换库的 **w**（写）权限。可以在第169页的『第18章 管理 CDS 目录』中读到 CDS 目录作用及其用途的详细信息。可以在第155页的『第16章 控制对 CDS 名称的访问』中读到关于 ACL 和 CDS 目录的更多信息。

在 DCE 单元中创建新的用户通常需要很多许可权，所以应当用单元管理员的帐户（或同样特权的帐户）登录。*IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference* 中列出了必需的许可权。

要在 DCE 单元中创建新的用户，调用 **user create** 操作。下列示例创建一个主体名称 **P_Pestana** 和一个同名的帐户。**create** 操作需要您输入口令，以避免其它人在无人照管的会话中创建一个未经授权的帐户。您也必须提供 **-password** 选项为该用户指定口令。必需的 **-group** 和 **-organization** 选项将主体 **P_Pestana** 添加到指定的组和组织中。可选的 **-fullname** 选项创建一个全名，以帮助其它用户认识这个主体。

```
dcecp> user create P_Pestana -fullname {Patricia Pestana} \
       -mypwd mxyzptlk -password change.me -group users \
       -organization managers
dcecp>
```

通过将一个用户名列表指定为 **user create** 操作的参数，可以创建多个用户。不过这种方法带来某些限制。所创建的所有用户将有相同的初始口令、组名和组织名，而且不能指定 **uid** 属性，因为这对每个用户都是唯一的。下列示例创建了多个用户，口令为 **change.me**，组名为 **users**，组织名为 **staff**:

```
dcecp> user create {R_Lee B_Joy N_Lynn D_Dee} -mypwd mxyzptlk \
       -password change.me -group users -organization staff
dcecp>
```

显示用户信息

有时您可能需要查看用户的属性。例如，可能要查看一个或多个帐户的失效日期，或查看某个主体的全名。

user show 命令显示与由参数指定的用户所关联的属性。属性包含主体属性和 ERA，帐户属性和策略。返回的信息就象按下列次序运行下列命令:

- **principal show**
- **account show -all**

下列命令显示与用户 **P_Pestana** 关联的主体和帐户属性:

```
dcecp> user show P_Pestana
{fullname {Pat Pestana}}
{uid 5139}
{uuid 00001413-ad4f-21cd-8c00-0000c08adf56}
{alias no}
{quota unlimited}
{groups users}
{acctvalid yes}
{client yes}
{created /.../my_cell.goodco.com/cell_admin \
    1994-08-01-16:41:32.000+00:00I-----}
{description {}}
{dupkey no}
{expdate none}
{forwardabletkt yes}
{goodsince 1994-08-01-16:41:32.000+00:00I-----}
{group users}
{home /}
{lastchange /.../my_cell.goodco.com/cell_admin \
    1994-08-01-16:41:32.000+00:00I-----}
{organization managers}
{postdatedtkt no}
{proxiabletkt no}
{pwdvalid yes}
```

```
{renewabletkt yes}
{server yes}
{shell {}}
{stdtgtauth yes}
nopolicy
dcecp>
```

通过将一个用户名列表指定为 **user show** 操作的参数，可以显示多个用户的信息。

删除用户

当用户脱离组织后，可能需要从单元中删除该用户。使用 **user delete** 命令可以完成这个工作。这个操作从注册表中删除主体名称，然后删除帐户，并且从任何组和组织中删除该主体。这个操作同样从 CDS 中删除 **/.:/users/***principalname* 目录和任何内容。

删除一个用户需要多种许可权，例如那些通常和单元管理员关联的许可权。请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference*。

下列示例从单元中删除用户 **P_Pestana**:

```
dcecp> user delete P_Pestana
dcecp>
```

通过将一个用户名列表指定为 **user delete** 操作的参数，可以删除多个用户。

```
dcecp> user delete {W_Rosenberry J_Hunter P_Pestana}
dcecp>
```

如果在外部单元中有许可权，通过指定用户的全局主体名，可以从中删除一个或多个用户。例如:

```
dcecp> user delete /.../their_cell.goodco.com/J_Jones
dcecp>
```

修改或扩展用户对象

user 任务对象是以脚本实现的，因此管理员可以根据现场的情况对其进行修改或扩展。例如，管理员可能需要在对象中添加 DFS 信息。下列包含其它可能的修改:

- 更改为用户创建或完全删除的 CDS 目录的位置。
- 更改对各种对象的缺省 ACL。
- 添加选项以赋予用户对 **/.:/users/***username* 所驻留的主副本中央交换库的写权限，这允许用户创建他们自己的子目录。这个选项可以将单独的主体名添加到中央交换库 ACL。一个更简单的方法是将主体添加到一个对中央交换库有写访问权的组。
- 对新创建的所有主体和帐户设置某些属性或策略以符合站点的策略。例如，可以将主体设置为带有一个 **pwd_val_type** 的 ERA，并设置帐户生成随机口令。
- 为口令（将由用户更改）、组、组织或主体目录等等设置站点特定的缺省值。
- 支持 **user modify** 命令。这样的命令可以更改组、组织信息或其它和用户关联的属性。

本指南 第1页的『第1部分 DCE 控制程序』部分讨论了用 **dcecp** 语言创建新 **dcecp** 对象或修改现存对象的方法。

第8章 事件管理服务 (EMS)

EMS 管理 DCE 单元中的事件服务。在 EMS 中，事件是从事件供应者发送到 EMS，再从 EMS 发送到一个或多个事件消费者的数据。事件包含一个事件头和一个包含事件特定类型数据的事件属性列表。

EMS 由三个主要组件构成:

- **EMS 守护程序 (emsd)** 是一个服务器，它:
  - 认证和授权事件供应者和消费者
  - 维护事件类型、事件过滤器和消费者的数据库
  - 将事件过滤器组与每个事件消费者相关联
  - 确保将事件可靠地传递到感兴趣的消费者。
- **事件供应者**是发出事件数据的任何基于 DCE 的用户应用程序。
- **事件消费者**是请求者，它:
  - 向 EMS 请求所支持的事件类型
  - 获得一个现有过滤器名称的列表
  - 对每种事件类型构造事件过滤器
  - 将是事件过滤器添加到其事件过滤器组。

EMS 数据结构被分成以下功能组:

- 事件
- 事件结构
- 事件类型
- 事件过滤器
- 消费者数据结构
- 服务器数据结构

将事件传递到 EMS 的能力与 SVC 和审计集成在一起。还可以使用 EMS 将事件通过 SVC 传递到 DCE SNMP 子代理。有关传递消息的详细信息，请参阅 DCE Application Development Guide --Core Components --SVC and Audit Sections。

DCE 管理功能包括 EMS 服务器、事件队列和事件日志的管理。有关详细信息，请参阅:

- 第72页的『启动 EMS 服务器』
- 第72页的『记录 EMS 事件』
- 第73页的『管理 EMS 消费者』
- 第74页的『管理 EMS 事件过滤器』
- 第75页的『管理 EMS 事件队列』
- 第75页的『管理 EMS 守护程序』
- 第76页的『设置 EMS 服务器的许可权』
- 第77页的『事件类型安全性管理』
- 第78页的『事件过滤器安全性管理』

- 第78页的『消费者安全性管理』
- 第79页的『EMS 安全性初始化』

启动 EMS 服务器

emsd 命令启动 EMS 守护程序。在消费者可以检索到事件或者供应者可以提供事件之前， EMS 守护程序必须在 DCE 单元中的主机系统上运行。

EMS 守护程序运行在本地主机机器主体标识下 (**host/***hostname***/self**)。当 **emsd** 启动时，DCE 主机守护程序 (**dced**) 必须在本地主机上运行。**emsd** 命令还需要 CDS Advertiser (**cdsadv**)。

emsd 命令有以下可选参数:

-l *log_directory*
指定日志文件驻留位置。

-q *queue_size*
指定 EMS 所计数的最大事件数。

-w *svc_route*
指定 DCE 可服务性路由指示。

要启动 EMS 守护程序并指定队列大小和日志位置，输入:

```
emsd -q 2048 -l /opt/dcelocal/var/ems
```

要启动 EMS 守护程序，指定可服务性传递指示，并定义最大队列大小，请输入:

```
emsd -w NOTICE:STDOUT:- -w NOTICE_VERBOSE:STDOUT-:-
```

记录 EMS 事件

EMS 事件日志用来在出现 EMS 故障时存储事件。EMS 将所有事件写入事件日志，并在将事件记录发送到所有预期消费者之后删除事件记录。事件日志保存在 **emsd** 所运行机器上的一个文件中。事件保存在由环境变量 **EMS_EVENTLOG_DIR** 所指定的目录中。提供了一个 API 来检查本地事件日志。

emslog 对象代表 EMS 事件日志。**emslog** 命令在 **dcecp** 环境中发出，后面跟有这些子命令之一:

help 返回有关对象的帮助信息。

operations
返回对象所支持的操作列表。

show 返回一个事件日志文件中的事件列表。

要显示常规 EMS 日志帮助信息，输入:

```
dcecp> emslog help
```

EMS 显示:

```
help        Print a summary of command-line options.
operations  Returns the valid operations for command.
show        Returns a list of events in the event log file.
```

要获得由对象支持的操作列表，输入:

```
dcecp> emslog operations
```

EMS 显示:

```
show help operations
```

要显示事件日志文件中的事件列表，输入:

```
dcecp> emslog show
EMS 显示:
--- Start of an EMS event record ---
Type: SVC:Event Id: 8d1b0b00-e9e7-11ce-8af3-10005a890435
Name Service: DCE /.../eagle_dce/hosts/hidalgod.austin.ibm.com
Description Name: EMS_Test_Producer
PID: 565 UID: 0 GID: 0
Severity: NOTICE
Arrival Time: 1995-09-08-14:06:32.970+00:00I-----
Printing 16 items
Item 1: [version] = ulong int 1
Item 2: [t] = 1995-09-08-14:06:32.970+00:00I-----
Item 3: [argtypes] = char string
Item 4: [table_index] = ulong int 0
Item 5: [attributes] = ulong int 64
Item 6: [message_index] = ulong int 389738500
Item 7: [format] = char string Test Supplier starting
Item 8: [file] = char string supplier.c
Item 9: [progname] char string EMS_Test_Producer
Item 10: [line] = ulong int 63
Item 11: [threadid] = ulong int 2
Item 12: [component_name] = char string sup
Item 13: [sc_name] = char string general
Item 14: [attribute.debug] = ushort int 0
Item 15: [attribute.severity] = ushort int 4
Item 16: [attribute.actroute] = ulong int 0
--- End of an EMS event record ---
```

管理 EMS 消费者

EMS 消费者注册向事件服务器注册来接收事件。每个消费者都有一个名称、一个 UUID、一个它所运行的主机和一列构成过滤器组的过滤器名称。

如果消费者进程异常终止，则系统管理员可能要使用 **emsconsumer** 命令删除消费者。

如果已经在过滤器数据库中定义了新的过滤器，则管理员可以使用 **emsconsumer** 命令来将那些过滤器添加到消费者过滤器组，以进一步定义消费者接收的事件。另外，如果消费者没有接收到它应该接收到的所有事件，则管理员可以用 **emsconsumer** 命令从消费者过滤器组删除过滤器。

DCE 提供 **emsconsumer** 命令和相关子命令来管理消费者。该命令在 **dcecp** 环境中发出，并可以执行以下子命令:

catalog
返回向主机上的 EMS 注册的消费者列表。

delete 从主机上的 EMS 删除已注册的消费者列表。

help 显示有关对象的帮助信息。

modify
修改有给定消费者相关的事件过滤器组。

operations
返回对象所支持的操作列表。

show 返回消费者过滤器组中的过滤器名称列表。

以下是必需的许可权:

- 对于 **emsconsumer catalog** 和 **emsconsumer show**，必须具有 **/.:/***hostname***/ems-server/consumers** 上的 **r** 许可权。
- 对于 **emsconsumer delete**，必须具有 **/.:/***hostname***/ems-server/consumers** 上的 **d** 许可权。
- 对于 **emsconsumer modify**，必须具有 **/.:/***hostname***/ems-server/consumers** 上的 **w** 许可权。

要获得注册到 EMS 上的消费者列表，输入:

```
dcecp> emsconsumer catalog
```

EMS 显示:

```
{consumer1 7e383761-f41f-11ce-9051-08005acd43c6 /.:/hosts/eagle.austin.ibm.com}
{consumer1 a4c7ff26-f449-11ce-a863-10005a4f3556 /.:/hosts/eagle.austin.ibm.com}
{consumer2 283cc40c-f447-11ce-9dd3-10005a4f3556 /.:/hosts/umesh.austin.ibm.com}
```

要将过滤器 **foo** 添加到 **consumer2** 事件过滤器组，输入:

```
dcecp> emsconsumer modify consumer2 -add {filter foo}
```

要显示 **consumer2** 过滤器组中的过滤器名称列表，输入:

```
dcecp> emsconsumer show consumer2
```

EMS 显示:

```
{foo2 foo3 foo4 foo5}
```

管理 EMS 事件过滤器

EMS 对从供应者接收到的事件应用 EMS 事件过滤器，以确定是否将事件转发到消费者。

EMS 事件过滤器是一个或多个过滤器表达式的集合。每个过滤器表达式都由属性名称、属性运算符和属性值组成。

可以在 **dcecp** 环境中发出 **emsfilter** 命令和相关子命令来管理本地主机上的事件过滤器。还可以指定 **--host** 选项来将命令发送到不同的主机。可以接受的 DCE 主机名称格式或者是整个 DCE 名称（例如，**/.:/hosts/jurassic.austin.ibm.com**）或者是带有域名称的 DCE 主机名（例如，**jurassic.austin.ibm.com**）。

emsfilter 执行以下子命令:

catalog
返回 EMS 中所有过滤器名称列表。

delete 从 EMS 中删除过滤器及其相关过滤器表达式。

help 显示有关对象的帮助信息。

operations
返回对象所支持操作的列表。

show 返回指定过滤器中的过滤器表达式列表。

以下是必需的许可权:

- 对于 **emsfilter catalog** 和 **emsfilter show**，必须具有 **/.:/***hostname***/ems-server/filters** 上的 **r** 许可权。
- 对于 **emsfilter delete**，必须具有 **/.:/***hostname***/ems-server/filters/***filtername** 上的 **d** 许可权。

要显示由 EMS 守护程序保留的过滤器，输入:

```
dcecp> emsfilter catalog
```

EMS 显示:

```
Filter1
Filter2
```

要删除名为 **Filter1** 的过滤器及其相关过滤器表达式，输入:

```
dcecp> emsfilter delete Filter1
```

要显示对象所支持的操作列表，输入:

```
dcecp> emsfilter operations
```

EMS 显示:

```
catalog delete show help operations
```

要在 **Filter2** 过滤器中显示表达式列表，输入:

```
dcecp> emsfilter show Filter2
```

EMS 显示:

```
{event_type == SVC}
{file == file.c}
```

管理 EMS 事件队列

可以在启动 **emsd** 时使用 **-q** 选项或 **EMS_QUEUE_SIZE** 环境变量设置 EMS 事件队列大小。如果 EMS 开始接收**队列满**错误，则应该使用大一些的队列大小来重新启动守护程序。有关 **emsd** 启动的详细信息，请参阅第72页的『启动 EMS 服务器』。

使用配置 GUI 时，只可以使用环境变量来增加队列大小。

管理 EMS 守护程序

EMS 守护程序 **emsd** 负责:

- 管理事件供应者和消费者方面的事件 ACL
- 维护事件类型、事件过滤器和消费者的数据库
- 将事件过滤器组与每个事件消费者关联
- 确保将事件可靠地传递到感兴趣的消费者。

ems 命令及其相关子命令管理 DCE主机上的 EMS 守护程序。该命令在 **dcecp** 环境中发出，并可以执行以下子命令:

catalog
返回当前单元中 EMS 守护程序正在其上运行的所有主机。

help 返回有关对象的帮助信息。

operations
返回对象所支持的操作列表。

show 返回 EMS 守护程序的属性列表。

ems show 命令要求您具有 **/.:/***hostname***/ems-server** 上的 **r** 许可权。

要列出在当前单元中运行的所有主机，输入:

```
dcecp> ems catalog
```

EMS 显示:

```
/.:/hosts/eagle.austin.ibm.com
/.:/hosts/umesh.austin.ibm.com
```

要返回对象所支持的操作列表，输入:

```
dcecp> ems operations
```

EMS 显示:

```
catalog show help operations
```

要显示 EMS 守护程序的属性列表，输入:

```
dcecp> ems show
```

EMS 显示:

```
{eventlog_dir /opt/dcelocal/dce/var/ems}
{queue_size 5000}
```

设置 EMS 服务器的许可权

EMS 提供在 EMS 数据库中对数据进行安全操作。这包括“事件过滤器数据库”、“事件类型数据库”和“消费者数据库”中的消费者。 EMS 还提供供应者和消费者认证和授权以及事件数据的安全传输。

所有 ACL 都与 DCE 名称空间中的名称相关，并且 EMSD 服务器管理超过联结处的名称空间:

```
/.:/hosts/hostname/ems-server/
```

与这个对象相关的 ACL 控制着对在这个名称空间中注册的 EMSD 服务器的访问。与 *ems-server* 相关的许可权是:

表 1. EMSD 服务器许可权位:

许可权位	名称	描述
c	控制	修改服务器上的 ACL
r	读	读这个服务器的属性

表 1. EMSD 服务器许可权位: (续)

许可权位	名称	描述
s	停止	停止 EMS 服务器
w	写	修改这个服务器上的属性

在 EMS 服务器联结处下维护着三个安全性对象。它们代表的目录和数据库是:

event-types
数据类型数据库

filters 过滤器数据库

consumers
消费者数据库

这些数据库中的每一个都有与之相关的 ACL。

事件类型安全性管理

事件类型数据库用 DCE 名称空间中的以下名称表示:

`/.:/hosts/`*`hostname`*`/ems-server/event-types`

与该对象相关的 ACL 控制对该数据库的访问。与 *event-types* 相关的许可权是:

表 2. 事件类型数据库许可权位

许可权位	名称	描述
c	控制	修改事件类型上的 ACL
d	删除	删除事件类型模式
i	插入	添加事件类型模式
r	读	读事件类型模式的内容

可以按事件类型来授予 EMS 事件数据访问权限。可以通过修改以下 ACL 来授予给定事件类型的事件数据上的权限:

`/.:/hosts/`*`hostname`*`/ems-server/event-types/`*`event_type_name`*

其中，*event_type_name* 是在事件类型模式中出现的事件类型名称。SVC 事件可识别的名称是:

`/.:/hosts/`*`hostname`*`/ems-server/events/SVC`

与 *event_type_name* 相关的许可权是:

表 3. 事件类型许可权位

许可权位	名称	描述
c	控制	修改事件类型上的 ACL
d	删除	删除事件类型
r	读	读（消费）该类型的事件
w	写	写（提供）该类型的事件

供应者权利在向 EMS 发送第一个事件时验证，消费者权利在将事件转发给那个消费者之前验证。使用经认证的 RPC 来访问 EMS 供应者和消费者远程 API。

事件过滤器安全性管理

过滤器数据库由 DCE 名称空间中的以下名称表示:

```
/.:/hosts/hostname/ems-server/filters
```

与该对象相关的 ACL 控制对该数据库的访问。与过滤器相关的许可权是:

表 4. 过滤器数据库许可权位

许可权位	名称	描述
c	控制	修改过滤器上的 ACL
d	删除	删除事件过滤器
i	插入	添加事件过滤器
r	读	获得事件过滤器内容列表

要为每个事件过滤器授予事件过滤器访问控制权限。通过修改以下的 ACL 来授予给定事件过滤器的过滤器访问权限:

```
/.:/hosts/hostname /ems-server/filters/filter_name
```

其中，*filter_name* 是在对 **ems_filter_add** 调用上授予事件过滤器的事件过滤器名称。

与事件过滤器相关的许可权是:

表 5. 事件过滤器许可权位

许可权位	名称	描述
c	控制	修改事件过滤器上的 ACL
d	删除	删除事件类型过滤器
w	写	修改事件过滤器的内容

当消费者创建了一个事件过滤器，那个消费者主体自动接收到已创建的事件过滤器上的 **dwc** 许可权。

消费者安全性管理

“消费者数据库”由 DCE名称空间中的以下名称表示:

```
/.:/hosts/hostname/ems-server/consumers
```

与该对象相关的 ACL 控制对该数据库的访问。与消费者相关的许可权是:

表 6. 消费者数据库许可权位

许可权位	名称	描述
c	控制	修改消费者上的 ACL
d	删除	删除消费者
i	插入	添加（注册）消费者
r	读	列出消费者信息
w	写	修改包括其过滤器组的消费者

EMS 安全性初始化

当配置 EMS 时，缺省创建几个安全性组。这些组是 **ems-admin**、**ems-consumer** 和 **ems-supplier**。缺省许可权是:

```
/.:/hosts/hostname/ems-server object acl
              ems-admin               crws
              hosts/hostname/self     rws
              any_other               r
   /.:/hosts/hostname/ems-server/event-types object acl
              ems-admin               cri
              ems-consumer            r
              ems-supplier            ri
              any_other               r
    initial object acl (/.:/hosts/&cont;hostname/ems-server/event-types/event_type_name)
              ems-admin               cdw
              ems-consumer            r
              ems-supplier            w
   /.:/hosts/hostname/ems-server/filters object acl
              ems-admin               crdi
              ems-consumer            ir
              any_other               r
   initial object acl (/.:/hosts/&cont;hostname/ems-server/filters/filter_name)
              ems-admin               cdw
   /.:/hosts/hostname/ems-server/consumers object acl
              ems-admin               cdrw
              ems-consumer            irwd
              any_other               r
```

因为这些许可权是为 **ems_admin** 组设置的，所以，每个自动创建的新事件过滤器和事件类型都继承同样的许可权。

管理员可以将主体添加到这些组，并授予它们对单元中所有 EMD 的访问权限。如果希望实现更高安全性，则可以从各自的 ACL 除去组，并可以添加主体。

第3部分 DCE 主机和应用程序管理

第9章 管理 DCE 主机服务和主机数据

一些产生或维护单元范围信息的服务，例如 DTS、CDS 和 DCE 安全性服务注册表是集中化的。虽然所提供的服务在整个单元中都可用，但是服务器本身通常仅驻留在单元中几个选定的主机上。

其他 DCE 服务是普遍的；也就是说，它们驻留在 DCE 单元中的每个主机上。在每个 DCE 主机上运行的 DCE 软件提供了必不可少的服务，使本地客户机和服务器程序以可靠和安全的方式与远程客户机和服务器程序交互。因此，DCE 单元中的每个主机都具有本章第一部分中讨论的管理问题。

对 DCE 环境中主机操作必不可少的每个 DCE 主机维护本地数据。有时候，可能因为单元配置更改或添加了 DCE 功能或 DCE 应用程序，需要修改部分数据。本章的第二部分讨论如何使用 DCE 控制程序获取对此数据的远程、已认证访问。

当 DCE 操作由于某些原因无法成功，需要通知适当人员发生的情况。 DCE 服务功能消息传递设施根据消息的严重性级别允许您路由错误消息。本章最后一部分说明了如何管理此设施。

DCE 主机服务

一些 DCE 主机服务（例如运行时库）是惰性的，且当在主机上配置了 DCE 后不需要任何管理。但其它服务是活动的程序。类似这样的活动服务为*端点映射器*，作为主机上的查表服务。端点映射器在主机*端点映射表*中列出服务器通信端口（称为*端点*）。寻找特定服务器的远程客户机查询端点映射器，返回端点映射表中包含的信息。端点映射器和其它活动的服务一起包含在称为 *DCE 主机守护程序*或 **dced** 的单一程序中。一般，当使用 DCE 软件配置主机时，主机引导进程将与其它守护程序或进程一起启动 **dced** 进程。但是，有时候可能需要手工启动或重新启动此守护程序。

dced 程序包括一套满足主机系统上 DCE 客户机和服务器应用程序很多需求的 DCE 主机服务:

- 端点映射器服务作为服务器目录在主机上运行。客户机可以通过查询主机端点映射表来获取已注册服务器的通信端点。
- 安全性确认服务管理本地主机上的 DCE 安全性。
- 服务器配置和执行服务让管理员远程设置服务器的启动和停止条件，明确地启动和停止个别服务器，并监视正在运行的服务器的状态。
- 密钥管理服务允许管理员远程地管理服务器口令。
- 主机数据服务允许管理员远程地管理主机文件中存储的数据。管理员将会发现这种远程管理主机单元名称和单元别名信息的方式非常有用。
- 属性模式功能允许管理员将新的属性添加到服务器配置信息中。

通常，管理 DCE 服务器（例如 DCE 单元目录服务器）或运行使用认证的基于 DCE 应用程序服务器或客户机的任何系统都必须同时运行 **dced** 进程。

因此，如果 **dced** 进程由于某些原因失败了，它将停止该进程的所有组件服务，使主机无法响应客户机的请求。类似地，如果 **dced** 进程由于某些原因异常退出，可能会造成一个组件服务（例如密钥管理服务）失败。当发生问题时，**dced** 和它的组件服务之间的这种关系是很值得了解的。

启动和停止 DCE 主机服务

虽然 **dced** 进程通常作为主机引导进程的部件启动，有时候仍需手工启动进程。

在启动 **dced** 之前，任何客户机／服务器通信依赖的基本网络服务都必须可用；例如，在大多数 UNIX 系统上必须启用网络接口和路由服务。一旦建立了这些传输层服务，就能启动 **dced**。当启动 **dced** 后，就可以启动基于 RPC 的服务器。

端点映射器在特权或保留通信端口（公认端点）上侦听客户机服务请求。因此，**dced** 必须以特权用户启动。

dced 部分（端点映射表）包含客户机用于定位主机系统上服务器的信息。**dced** 进程在称为 *dcelocal***/var/dced/Ep.db** 的数据库文件名中维护此信息的副本，因此，如果由于某些原因停止然后重新启动 **dced**，则不会丢失它。另一个称为 *dcelocal***/var/dced/Srvrexec.db** 的数据库文件维护当前正在主机上运行的服务器（例如每个服务器的进程标识）信息。因为大多数服务器在每次启动时获取不同的端点和不同的进程标识，所以当系统重新引导时，这些数据库中的信息将过时。

可以配置 **dced** 以便每次主机引导时启动。要做到这点，请使用 **config.dce** 上的 **-autostart yes** 选项。当启动任何其它 DCE 守护程序时，**Start.dce** 都将启动 **dced**，除非当没有 **dced** 时已配置了瘦客户机。

通常不需要在外壳中启动 **dced**，如果需要这样做，只要以 **root** 登录，并输入下列命令：

```
# dcelocal/bin/dced
```

缺省时，**dced** 为主机支持并在该主机上运行的每次传送侦听一个端点。也就是说，如果主机既支持 TCP/IP，又支持 UDP/IP 传送，**dced** 将根据客户机请求在 TCP 和 UDP 套接字上侦听。可选的 *protseq* 变量允许您将 **dced** 使用的传送限制为您所指定的传送。为了获得预期的调试能力，此功能应该慎用；如果限制传送，客户机将无法在已排除传送上定位服务器，服务器也无法使用已排除传送将它们注册到端点映射表中。如需有关可选的 *protseq* 变量的信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference*。

虽然应该在 DCE 客户机或服务器应用程序运行的所有主机上运行主机服务，但也存在一些情况可以避免运行它们。

- 不执行认证的 DCE 客户机
- 不执行认证且不使用端点映射器或其它活动的 DCE 主机服务的 DCE 服务器。

当启动了 DCE 主机服务时，可以通过使用 DCE 控制程序 **dcecp** 执行所有的 DCE 主机和服务器管理任务。控制程序提供对主机和服务器管理功能安全、远程访问，这意味着您可以管理所有 DCE 主机，无需登录到每个主机。本书第1页的『第1部分 DCE 控制程序』说明如何以交互方式使用 **dcecp**，以及如何编写 **dcecp** 脚本以管理 DCE 活动。在执行本章或本文档其它地方说明的管理任务之前，应该了解那些基本知识。

DCED 主机守护程序的异常终止

如果 DCE 守护程序异常停止或退出，可以重新启动它。重新启动的 **dced** 进程不会丢失任何以前已注册的服务器绑定。如果简单地从 **Ep.db** 和 **Srvrexec.db** 文件装入信息。作为规则，不建议停止和重新启动 **dced**，因为它同时也停止安全性确认服务。

如果 **dced** 异常停止或退出了，某些文件可能被毁坏，并需要在重新启动 DCE 之前删除和重新启动它们。在这种情况下，请执行下列操作:

1. 在命令行输入 **stop.dce all**，停止已配置 DCE 组件的所有守护程序。
2. 输入 **clean_up.dce** 以删除可能被毁坏的文件。如果需要，可以重新创建所有被删除的文件。
3. 输入 **start.dce all** 重新启动所有已配置的组件。

管理主机数据

DCE 单元中的每个主机都维护 DCE 环境中操作必不可少的本地数据。例如，每个主机 DCE 标识都依赖于某个数据项，该数据项指定主机名、单元名称和任何单元别名。当前，这些数据项被存储在称为 *dcelocal*/**dce_cf.db** 的本地文件中。可以使用 DCE 控制程序的 **hostdata** 对象来远程地修改这些和其它数据项。

hostdata 对象还具有许多更广泛的应用程序；管理员将发现使用安全和平台独立方法访问远程主机上的常规数据和文件非常有用。本章最后一部分检查此功能强大的访问方法。

访问主机数据的许可权

访问控制表 (ACL) 防止未授权的主体创建、更改或删除主机数据信息。两种类型的 ACL 保护 hostdata 信息。一种类型 ACL 保护 hostdata 项驻留的容器。第二种类型保护每个单独 hostdata 项。

这部分显示如何管理保护 hostdata 信息的 ACL。有关设置和使用 ACL 保护的详细信息，请参阅第261页的『第28章 使用访问控制表』。

Hostdata 容器的许可权

在 DCE 中，hostdata 项驻留在容器中，它实际上是 **dced** 维护的支持存储器机制。在 UNIX 系统上，它通常是一个称为 *dcelocal*/**var/dced/Hostdata.db** 的文件。文件由 root 拥有，它的经由 **dced** 的访问受 ACL 保护。这些 ACL 许可权控制能访问容器中数据的用户。每个 DCE 主机都具有使用下列名称的 hostdata 容器 ACL:

/.../*cellname*/**hosts**/*hostname*/**config**/**hostdata**

Hostdata 容器 ACL 具有下列许可权:

c （控制）
修改容器 ACL。

r （读）
读取容器中的 hostdata 项列表。

i （插入）
创建新的 hostdata 项。

I （插入）

虽然存在 **I** 许可权，它并不应用到 hostdata 项。许可权应用到服务器控制设施，该设施在 第93页的『第10章 DCE 应用程序管理』 中说明。

使用 **dcecp acl** 对象查看或修改 ACL。例如，使用下列操作查看主机 **silver** 上 Hostdata 容器对象的 ACL:

```
dcecp> acl show /.:/hosts/silver/config/hostdata
{user hosts/silver/self criI}
{unauthenticated r}
{any_other r}
dcecp>
```

Hostdata 项的许可权

下列每个主机标识数据项都受 ACL 保护:

/.../*cellname***/hosts/***hostname***/config/hostdata/host_name**
/.../*cellname***/hosts/***hostname***/config/hostdata/cell_name**
/.../*cellname***/hosts/***hostname***/config/hostdata/cell_aliases**
/.../*cellname***/hosts/***hostname***/config/hostdata/post_processors**

每个 ACL 都具有下列许可权:

c （控制）
修改 ACL

d （删除）
删除项

p （清除）
删除项的支持存储器

r （读）
读取项的数据

w （写）
修改项的数据

使用 **acl** 对象查看或修改 ACL。例如，使用下列操作查看主机 **silver** 上 **cell_aliases** hostdata 的 ACL:

```
dcecp> acl show /.:/hosts/silver/config/hostdata/cell_aliases
{unauthenticated ---r-}
{user hosts/silver/self cdprw}
{any_other ---r-}
dcecp>
```

修改主机单元名称信息

使用 **hostdata** 对象，可以添加、更改和删除 DCE 主机上的数据项。管理员将发现这对修改主机的单元名称或单元别名信息非常有用，他们还可以在可从主机访问的其它数据上操作。

每个 DCE 主机维护其中已注册了主机的单元名称和单元别名的受保护本地副本。主机将此信息保存在 **root** 拥有的称为 *dcelocal***/dce_cf.db** 的本地文件中。主机将此信息用于认证目的-作为主机标识信息的一部分。

虽然主机单元名称信息相对比较稳定，仍存在需要更改此信息的环境:

- 当主机移动到不同的单元时

- 当主机单元名称更改了或单元名称需要别名

但是，当出现这些情况中的任一种时，仅在主机上更新单元名称信息是不够的。单元名称信息必须同时在 CDS 和 DCE 安全性服务注册表中更新。由于这些目的，**dcecp** 提供 **cellalias** 任务对象，它在需要更改时将更新单元名称信息。

当主机移动到不同的单元时，应该运行 **unconfig.dce** 从一个单元删除主机。然后运行 **config.dce** 在新的单元中建立主机。

当主机的单元名称更改或单元名称获取别名时，应该执行 **cellalias** 操作，该操作更新 CDS、DCE 安全性服务注册表和单元中每个受影响主机的 **dce_cf.db** 文件中的单元信息。

但是，有时较高级别 **dcecp** 任务对象不提供例如当在某处修正被毁坏的文件或由于某些原因手工配置主机时可能需要的足够的控制。在这些情况中，可以使用 **hostdata** 对象更改个别主机上的单元名称信息。

然而请注意，这种 **hostdata** 对象的使用通常是一种故障检测操作，取决于主机单元信息何时与存储在 DCE 注册表或 CDS 中的其它单元信息不同步。这种情况可能是单元中使用很多主机的公共情况。

要更新主机是行的单元名称或单元名称信息，请使用 **hostdata** 对象。下列示例在称为 **/.../my_cell.goodco.com** 的单元中编制 **hostdata** 对象目录。然后，显示主机 **silver** 上的 *cell_name* 的内容。最后，修改单元名称为主机 **silver** 上的 **/.../my_cell.goodco.com**。

```
dcecp> hostdata cat
/.../my_cell.goodco.com/bronze/config/hostdata/dce_cf.db
/.../my_cell.goodco.com/bronze/config/hostdata/cell_name
/.../my_cell.goodco.com/bronze/config/hostdata/host_name
/.../my_cell.goodco.com/bronze/config/hostdata/cell_aliases
/.../my_cell.goodco.com/bronze/config/hostdata/post_processors
dcecp> hostdata show cell_name
{uuid 00174f6c-6eca-1d6a-bf90-0000c09ce054}
{annotation  {Name of cell}}
{storage cell_name}
{data {/.../old_cell.goodco.com}}
dcecp> hostdata modify \
       /.../my_cell.goodco.com/hosts/bronze/config/hostdata/cell_name \
       -data {/.../my_cell.goodco.com}}
dcecp>
```

在其它主机文件中处理数据

当 **hostdata** 对象可用于更改单元名称和单元别名信息时，它还具有更广泛的用途；可以使用它从可在 DCE 主机上访问的任何文件添加、更改和删除数据。

一个有用的示例是添加一个新的 CDS 属性。每个 DCE 主机都具有它自己的 CDS 属性文件 (**cds_attributes**)，它在其中为每个 CDS 属性存储了对象标识。可以使用本地主机编辑器来添加属性，然后将新的文件复制到每个主机。但这种方法需要您登录到每个主机。一个简单的方法是使用 **hostdata** 对象将新的属性添加到 CDS 属性文件。将操作放入 **foreach** 循环中，该循环将为单元中的每个主机重新执行该操作。

1. 使 CDS 属性文件作为 **hostdata** 对象的一个对象可访问。首先，使用 **hostdata** 对象创建一个表示 CDS 属性文件的 CDS 项。将存储器属性设置为 CDS 属性文件的主机文件名。下列示例假设 CDS 属性文件位于缺省位置，且文件存在:

```
dcecp> hostdata create /.:/hosts/silver/config/hostdata/cds_attr \
> -storage /opt/dcelocal/etc/cds_attributes -entry
dcecp>
```

2. **hostdata** 对象通过将文件中所有数据替换为您指定的新的数据来修改文件中的数据。下列示例显示完成此操作的一种方法。首先，检索并将所有行存储为变量中的 **dcecp** 列表元素。然后，通过使用 **attrlist** 命令创建一个新的变量，以将新行作为列表元素添加到变量中。最后，将新的变量复制回文件。

```
dcecp> set val [attrlist getvalues [hostdata show /.:/hosts/silver/config/ \
hostdata/cds_attr] > -type hostdata/data]
dcecp> set newval [attrlist add $val -member {NEW_ATTR 1.2.3.4}]
dcecp> hostdata modify /.:/hosts/silver/config/hostdata/cds_attr -data
$newval
dcecp>
```

路由可服务性消息

DCE 可服务性机制设计为主要用于服务器信息和错误消息--也就是说，与服务器维护和管理（以这些术语的广义）有关的人员感兴趣的消息。机制的主要概念是所有对维护或恢复常规操作来说重要的服务器事件都必须以自我记录的消息方式报告。结果（假设所有事件都已被正确标识和报告了），用户和管理员将总能了解在给定情况中他们应该采取什么操作。

注: 用户提示、交互式、客户机生成的消息传递是通过标准 DCE 消息传递接口处理的。

可服务性组件由 DCE 组件（RPC、DTS、安全性等等）用于它们自己的服务器消息传递，它可作为 API 由希望标准化应用程序的服务器消息传递的 DCE 应用程序员使用。（可服务性 API 在 *IBM DCE Version 3.2 for AIX and Solaris: Application Development Guide--Core Components* 中描述。）

消息传递使用 XPG4 (*X/Open Portability Guide*) 信息目录保持信息正文，但它将附加的层添加到 XPG4 功能。消息目录和其它必需数据（和文档）文件由称为 **sams**（符号和消息字符串）的实用程序生成。它的输入是文本文件，该文本文件建立一些有关要使用消息的程序的组织化信息，后随一系列消息本身的规范说明。可服务性机制允许系统管理员控制这些消息的路由。具体来说，可以根据为消息定义的严重性级别（**FATAL**、**ERROR** 等等）来定义消息路由。

下列部分描述如何控制可服务性消息的路由。首先，提供您 DCE 中可服务性消息传递的概述。然后，文本描述如何使用消息严重性级别来控制路由。最后，描述可以为可服务性消息指定路由的不同方式。

可服务性消息严重性级别

可服务性消息由*严重性级别*分类，该级别提供了有关导致程序发出消息的情况的重要信息。每个消息的严重性都由消息文本自身定义（例如，**通知**表明消息是信息性通知），系统管理员可以根据它们的严重性级别来不同地路由消息。

表7列出可能的严重性级别，并为每种级别提供说明。

表 7. 可服务性适消息严重性级别

名称	含义
FATAL	严重错误退出：发生不可纠正的错误（例如数据库崩溃），可能需要手工介入改正。程序通常在这种错误后立即终止。

表 7. 可服务性适消息严重性级别 (续)

名称	含义
ERROR	检测到错误: 发生非终止性异常事件（例如超时），或可由人工介入改正。虽然有些功能或服务可能无法再用，程序仍将继续操作。这种严重性级别也可用于表明无法完成特定的请求或操作。
WARNING	可纠正错误: 发生可自动纠正的错误（例如，没有发现配置文件，则使用缺省值）。如果效果不是所期望的，也可使用此严重性级别来表明*可能*为错误的条件（例如，删除非空目录的副作用是删除所有的文件）。此严重性级别也可用于表明如果不纠正的话将最终导致错误的情况（例如，打印机在纸上运行速度很慢）。
NOTICE	信息性通知: 发生了重要的日常事件; 例如，已启动服务器。
NOTICE_VERBOSE	详细信息通知: 发生了重要的日常事件; 例如，已删除目录项。

如何路由可服务性消息

可将可服务性适消息写入任何正常的输出目的地。可以按下列四种方式之一为可服务性适消息指定路由:

- 如果服务器支持远程可服务性适接口，则通过 **dcecp 日志**对象
- 按路由文件内容
- 按环境变量内容
- 如果受服务器支持，则按命令行标志（通常是 **-w**）

注: 每种方法都接受可服务性适路由规范的字符串语法格式。此外，**dcecp** 还允许您使用 Tcl（工具命令语言）语法，当编写脚本时易于使用。

*路由*消息实际包含指定两件事:

- 应该如何处理消息（即，它应放入的表单）
- 消息应该发送到的地方（它的目的地）

这两个规范有时紧密相关，有时指定某个目的地暗示消息必须放入某个表单。这允许缩写某些组合。

路由可服务性适消息的方法在下列部分中分别描述。

使用 dcecp log 对象

dcecp log 对象表示给定服务器的 DCE 可服务性适消息的当前路由状态。**log** 对象同时支持可服务性适路由和调试路由。

log 对象导出一系列操作。下列操作对可服务性适消息路由有用:

- **log list** 操作返回服务器注册的组件列表。 **-comp** 选项允许您为一个或多个命名组件返回子组件列表。
- **log show** 操作返回描述服务器的当前可服务性适路由规范的列表。
- **log modify** 操作为一个或多个指定服务器设置消息路由规范。

有关 **dcecp log** 对象的完整描述和它支持操作的语法，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference*。

此章节的剩余部分仅描述 **log modify** 操作以及如何使用它以建立可服务性适消息的路由。记住，路由总是按照每服务器设置，并为每个服务器记录在 **log** 对象中。

log modify 操作的语法为:

```
log
modify {string_binding_to_server | RPC_server_namespace_entry} \
    {-change serviceability_routing_specifications}
```

可以将多个目标服务器指定为隔离空格列表。通过补充描述服务器网络位置 (*string_binding_to_server*) 的 RPC 字符串绑定或服务器 (*RPC_server_namespace_entry*) 的名称空间项来指定每个服务器。当指定多个服务器时，可以在相同的列表中混合表单。

serviceability_routing_specification 是可服务性适路由元素的隔离空格列表。在个别路由选择元素的规范说明中不允许有空格。每个路由元素都是由包含可移植字符集 (PCS) 数据的四个字段组成的子串，如下所示（以字符串语法格式显示）:

severity:*output_form*:*destination*[:*application-defined*]

其中:

severity
: 消息严重性级别: **FATAL**、**ERROR**、**WARNING**、**NOTICE** 或 **NOTICE_VERBOSE**。

output_form
: 指定应该如何处理关联严重性级别的消息，这些消息必须是下列之一:

  BINFILE
  : 将这些消息作为二进制日志项编写

  TEXTFILE
  : 将这些消息作为可供人阅读文本编写

  FILE 等价于 **TEXTFILE**

  DISCARD
  : 不要记录这些消息

  STDOUT
  : 将这些消息作为可供人阅读文本编写到标准输出

  STDERR
  : 将这些消息作为可供人阅读文本编写到标准错误

  不要将 **NOTICE** 和／或 **NOTICE_VERBOSE** 消息的路由设置到 **STDERR**。这些消息应该路由到 **STDOUT**、**FILE**、**EMS** 等。如果 **NOTICE** 和／或 **NOTICE_VERBOSE** 消息路由到 **STDERR**，则 DCE 配置将失败。

  作为 **BINFILE** 编写的文件可以使用一套日志文件 API 来读取和操纵，在 *IBM DCE Version 3.2 for AIX and Solaris: Application Development Guide--Directory Services* 中有详细描述。

在 DCE 配置期间，不要将资料信息路由到 **STDERR**，因为配置代码可能将它们误解为错误。在配置期间，这些消息应该路由到一些其它文件，如缺省路由文件中所做的。

BINFILE、**TEXTFILE** 和 **FILE** *output_form* 指示符后面可能跟随表单的 2 位数字指示符:

.gens.count

其中:

gens 是指定应该保持的文件数的整数（即，代）

count 是指定应该将多少项（即，消息）写到每个文件中的整数

多个文件通过添加 **.**（点）到简单指定命名中，后随当前代号来命名。当文件中的项数达到 *count* 指定的最大数，文件将被关闭，代号递增，打开下一个文件。

当已经创建和填充了最大文件数，代号将复位回 **1**，具有该代号的新的文件将被创建并写入（这样使用相同的名称来覆盖已经存在的文件），依次类推。因此，文件回绕到开始处，且日志文件的总数不会超过 *gens*，虽然只要程序继续写它们，将继续写消息。

destination

指定消息应该发送到的地方，是一个路径名。如果指定的 *output_form* 是 **DISCARD**、**STDOUT** 或 **STDERR**，则可以将此字段留为空白。此字段还可以在文件名中包含一个 **%ld** 字符串，当写文件时，写消息的程序的进程标识将替换它。文件名*不能*包含 :（冒号）、;（分号）、% （百分号）或空格字符。

application-defined

用于特定应用程序信息。标准 DCE 程序忽略它。

字符串语法: 可服务性适路由规范的字符串语法是:

severity:*output_form*:*destination*[:*application-defined*][;...]

注意，可以将多路由选择规范定义为以分号分隔的列表。

例如，此规范:

```
FATAL:TEXTFILE:/dev/console;STDOUT:
ERROR:TEXTFILE.5.100:/tmp/errors
EXIT:DISCARD:
*:FILE:/tmp/svc-log
NOTICE:BINFILE:/tmp/log%ld
WARNING:STDOUT:
```

指示可服务性适机制执行下列操作:

- 将严重错误消息发送到控制台和标准输出
- 将其它错误消息发送到日志卷动文件
- 废弃正常退出报告
- 将所有消息写入日志文件
- 将资料信息发送到临时二进制日志
- 将警告信息发送到标准输出

Tcl 语法： 可服务性适路由规范的 Tcl 语法是:

{*severity output_form destination application-defined*}

其中，*severity*、*output_form*、*destination* 和 *application-defined* 如以前描述的所指定。在 Tcl 语法中，多路由选择规范采用下列表单:

{ {*specification*} {*specification*} {*specification*} }

例如，以前为字符串格式显示的样本规范将在如下 Tcl 语法中表示为:

```
{FATAL { {TEXTFILE
/dev/console} STDOUT} }
{ERROR TEXTFILE.5.100 /tmp/errors}
{EXIT DISCARD}
{* FILE /tmp/svc-log}
{NOTICE BINFILE /tmp/log%ld }
{WARNING STDOUT {} }
```

使用路由文件

如果存在称为 *dce-local-path***/var/svc/routing** 的文件，将使用文件（如果处于正确的格式）内容来确定可服务性适机制编写的消息路由。

可服务性适路由文件的缺省位置通常是 */opt/dcelocal***/var/svc/routing**。但是，可以通过将环境变量值 **DCE_SVC_ROUTING_FILE** 设置为完整期望的路径名来为文件指定不同的位置。

路由文件包含指定各种类型消息（基于消息严重性级别）期望的行。每行包含如下三个字段:

severity:*output_form*:*destination*[:*application-defined*][...]

可以同指定附加的 *output_form*:*destination* 对作为以分号分隔的列表来补充多路由选择。

在路由文件中，以 **#** 字符开始的空白行作为注释处理。

使用环境变量

可服务性适消息路由选择还可由特定环境变量的内容指定。如果使用环境变量，则指定的路由将覆盖任何路由文件指定的冲突路由。

可以通过将期望的路由指令放入下列环境变量中来指定（基于严重性级别）路由:

- **SVC_FATAL**
- **SVC_ERROR**
- **SVC_WARNING**
- **SVC_NOTICE**
- **SVC_NOTICE_VERBOSE**
- **SVC_BRIEF**

每个变量应该包含下列格式的单一字符串:

severity:*output_form*:*destination*[:*application-defined*][...]

可以同指定附加的 *output_form*:*destination* 对作为以分号分隔的列表来补充多路由选择。

第10章 DCE 应用程序管理

随着 DCE 的发展，普通需求的功能已被包含入 DCE 基础结构中。如例所示，DCE 包括能管理服务器操作和帮助服务器以受控和有效的方式退出的服务器控制功能。应用程序开发者可以基于这些能力，而非实施特殊的机制独立在每个服务器中处理它们。

将普通需求功能从应用程序移到 DCE 基础结构提供了重要的优点。应用程序能更小和更简便地开发和维护。更重要的是，因为应用程序不会被许多特殊的代码阻碍，因此更易使用不同类型的客户机来重新配置和重新连接。这个适应能力对组织努力跟上不断更新的商业需求非常重要。

DCE 应用程序总是具有管理方面。程序通常包含必需的功能以管理它们自己的管理需求，但这种方式对管理员来说可能很笨拙和不灵活。现在，实际上所有管理功能都能通过 **dcecp** 对程序员和管理员可用。这并不意味着程序员不再需要处理这些问题。我们希望一些程序员提供使用 **dcecp** 编写的脚本，配置客户机和服务器程序以在指定条件下启动和停止。

虽然这种方法提供了方便和一致的方式来管理应用程序，它同时也创建了编程和管理问题重叠的区域。本章中的讨论包括重叠区域和可能需要管理操作的注意环境。

控制服务器操作

DCE 应用程序服务器的普通观念假设服务器正在运行和等待客户机对服务的请求。这对某些普通服务器操作是一个有效的模型，但不提供 DCE 应用程序需要的灵活性。商业环境可能具有很多种类型的服务器。一些可能需要持续可用，另一些可能仅在一天的特定时间需要。其它一些在少见或不可预测基础上需要。

应用程序员或管理员可以通过编写监控服务器操作的脚本或应用程序，在需要时自动启动或重新启动服务器来解决这些类型的问题。这种解决方案通常依赖于类似启动和关机程序的主机实用程序或类似 **cron** 的调度程序。但是，这经常需要管理员登录到每个主机上的独立系统管理帐户。此外，此方法还给开发者和管理员带来更多负担，特别是在不同种类环境中，用在可能不可移植的设备无关服务器控制机制。

DCE 通过提供服务器控制设施（该设施提供各种方法来控制 DCE 应用程序服务器）来解决一部分这些问题。服务器控制设施是 DCE 守护程序 (**dced**) 的一部分，所以无论在何处运行 **dced**，服务器都能依赖它。而且，设施的管理功能可以经由 **dcecp** 访问，所以管理员能使用一致的（可移植）方法从可使用 **dcecp** 的任何主机管理服务器。此外，已认证了对服务器控制设施的访问，防止未授权的或服务器控制信息的意外篡改。

下列章节显示了一些公共配置需求，并描述配置和取消配置服务器的方法，以及如何启动和停止服务器和如何查看服务器信息。

公共服务器配置需求

在配置服务器之前，可能需要执行一些预备步骤。如果服务器使用 DCE 认证和授权，它的主体名称必须使用 DCE 安全性服务注册或在父代进程的 DCE 实体下运行。有关创建服务器帐户的详细信息，请参阅第311页的『第31章 创建和维护帐户』。

名称服务器配置信息

服务器配置信息可以使用以下名称形式来访问: **/.../***cellname*/**hosts/***hostname*/**config/srvrconf/***servername*。如果具有必需的许可权，可以使用全局名来访问远程主机（即使另一个单元中的主机）上的配置数据库。下列示例显示远程单元 **/.../their_cell.goodco.com** 中主机 **krypton** 上 **video_clip** 服务器的配置信息:

```
dcecp> server show /.../their_cell.goodco.com/hosts/krypton/config/srvrconf/video_clip
{uuid 2fa417e8-bb4c-11cd-831b-0000c08adf56}
{program {vclip}}
{arguments {-catalog}}
 .
 . （省略输出）
 .
dcecp>
```

下一个示例显示本地单元中主机 **silver** 上 **video_clip** 服务器的配置信息:

```
dcecp> server show /.:/hosts/silver/config/srvrconf/video_clip
{uuid 2fa417e8-bb4c-11cd-831b-0000c08adf56}
{program {vclip}}
{arguments {-catalog}}
 .
 . （省略输出）
 .
dcecp>
```

使用简名来显示本地主机上 **video_clip** 服务器的配置信息:

```
dcecp> server
show video_clip
{uuid 2fa417e8-bb4c-11cd-831b-0000c08adf56}
{program {vclip}}
{arguments {-catalog}}
 .
 . （省略输出）
 .
dcecp>
```

服务器配置信息

每个 DCE 都具有存储该主机上服务器的配置信息的数据库。使用 DCE 控制程序 **server** 对象来存储、修改或删除主机系统上服务器配置信息数据库中的服务器配置信息。

当管理服务器配置时，需要指定一些或所有下列信息:

uuid 特定服务器配置对象的标识符。

program
调用服务器程序的名称（包括路径名）。

directory
程序的工作目录名称。当服务器运行时，可能需要空间来存储它的输出或临时文件。

arguments
用于启动服务器的命令行变量。

entryname
服务器将它的绑定导出到的 RPC 项名称。

keytabs
服务器存储密钥的一个或多个相关 keytab 对象（文件）的 UUID。使用 DCE 认证或授权的服务器需要此信息。

principals
在 DCE 安全性服务中注册的服务器的一个或多个主体名称列表。使用 DCE 认证或授权的服务器需要此信息。

services
标识服务器提供的服务。每个服务属性包含具有下列元素的属性列表:

annotation
描述服务的供人阅读的字符串。

ifname
此服务的接口名称（在接口定义文件中指定）。

interface
此服务的*接口标识符*（UUID 和版本号）（在接口定义文件中指定）。

binding
标识此服务的字符串绑定列表。

entryname
服务器为此服务将绑定导出到的 RPC 项的名称。

flags 为此服务器标识标志的关键字列表。当前仅支持 **disabled** 标志。

objects
此服务支持的对象 UUID 列表。

uid 服务器与之一起启动的 POSIX UID。

starton
指定服务器启动条件。值是下列一个或多个列表:

auto 无论何时 DCE 守护程序接收到服务请求时，服务器也将启动。

explicit
无论何时管理员执行直接命名服务器的 **server start** 或 **server stop** 操作时，服务器也将启动或停止。

boot 无论何时主机系统启动时，服务器也将启动。

failure 无论何时服务器以失败退出状态退出时，服务器也将启动。

访问服务器控制设施的许可权

ACL 防止未授权主体创建、读取、更改或删除服务器控制设施维护的信息。

服务器控制设施维护两种服务器控制信息。服务器配置信息（在 DCE 中称为 **srvrconf**）由需要启动服务器的信息组成。服务器执行信息（在 DCE 中称为 **srvrexec**）包含服务器运行时需要控制或停止服务器的信息。

服务器配置信息受两种类型的 ACL 保护。一种类型 ACL 保护服务器控制信息驻留的容器。第二种 ACL 类型保护每个单独服务器的配置信息。

类似地，服务器执行信息受两种类型的 ACL 保护。一种类型 ACL 保护服务器执行信息驻留的容器。第二种 ACL 类型保护每个个别服务器的执行信息。

这部分显示如何管理保护服务器控制信息 ACL。有关设置和使用 ACL 保护的详细信息，请参阅第261页的『第28章 使用访问控制表』。

服务器配置容器的许可权： 服务器配置信息驻留在容器中。容器是一种在 UNIX 系统上作为文件实施的支持存储器机制，它由 root 用户拥有且受 ACL 保护。这些 ACL 许可权控制能访问容器中信息的用户。每个 DCE 主机都具有使用下列名称的服务器配置容器 ACL:

/.../*cellname*/**hosts/***hostname*/**config/srvrconf**

服务器配置容器 ACL 具有下列许可权:

c （控制）
修改容器 ACL。

r （读）
阅读容器中的配置信息。

i （插入）
创建新的配置信息。

I （插入）
为作为特权用户（例如，作为 POSIX 系统上的 root 用户）运行的服务器创建新的配置信息。这些操作还需要 **i** 许可权。

使用 **dcecp acl** 对象查看或修改 ACL。例如，使用下列操作查看主机 **silver** 上服务器配置容器对象的 ACL:

```
dcecp> acl show /.:/hosts/silver/config/srvrconf
{user appl_admin criI}
{unauthenticated r}
{any_other r}
dcecp>
```

因为 **/.:/hosts/silver/config/srvrconf** 是容器，它还具有“初始容器 ACL”和“初始对象 ACL”。可以通过对 **acl** 操作使用 **-ic** 和 **-io** 选项来操作这些初始 ACL。但是请注意，因为当前无法在 **/.:/hosts/***hostname***/config/srvrconf** 下创建子容器，因此“初始容器 ACL”没有效果。

访问服务器配置信息的许可权： 每个服务器的配置信息都受她自己的 ACL 保护。这些 ACL 可以防止未授权的主体创建、读取、更改或删除服务器配置信息，防止他们启动、停止、启用和禁用服务器。

每个 ACL 都是为它保护的服务器配置信息命名的，且具有类似下列的名称:

/.../*cellname*/**hosts/***hostname*/**config/srvrconf/***server_name*,

ACL 具有下列许可权:

c （控制）
修改 ACL。

d （删除）
删除服务器配置信息。

f （标志）
启动具有定制标志的服务器。

r （读）
读服务器配置信息。

w （写）
修改服务器配置信息。

x （执行）
启动服务器。

使用 **acl** 对象查看或修改 ACL。例如，使用下列操作查看主机 **silver** 上 **video-clip** 服务器的 ACL:

```
dcecp> acl show /.:/hosts/silver/config/srvrconf/video_clip
{user appl_admin cdfrwx}
{unauthenticated r}
{any_other r}
dcecp>
```

此 ACL 从容器的“初始对象 ACL”获取缺省值。可以通过对 **acl** 操作使用 **-io** 选项，在“初始对象 ACL”上操作。下列示例显示 **video_clip** 服务器的“初始对象 ACL”：

```
dcecp> acl show /.:/hosts/silver/config/srvrconf/video_clip
-io
{unauthenticated r}
{any_other r}
dcecp>
```

服务器执行容器的许可权： 当启动服务器时，DCE 守护程序将服务器配置信息复制到服务器执行数据库中。**dced** 进程还添加有关运行服务器（例如 UUID、服务器通信端点和它的进程名称和标识）的详细信息。执行信息控制运行服务器；例如，进程标识用于停止服务器。当服务器退出时，DCE 将删除它的服务器执行信息。

服务器执行信息驻留在容器中。容器是一种在 UNIX 系统上作为文件实施的支持存储器机制，它由 root 用户拥有且它的通过 **dced** 的访问受 ACL 保护。这些 ACL 许可权控制能访问容器中信息的用户。每个 DCE 主机都具有使用下列名称的服务器执行容器 ACL:

/.../*cellname*/**hosts**/*hostname*/**config**/**srvrexec**

服务器执行容器 ACL 具有下列许可权:

c （控制）
修改容器 ACL。

r （读）
阅读容器中的执行信息。

i （插入）
创建新的执行信息。

l （插入）
为作为特权用户（例如，作为 root 用户）运行的服务器创建新的执行信息。这些操作还需要 **i** 许可权。

使用 **acl** 对象查看或修改 ACL。例如，使用下列操作查看主机 **silver** 上服务器执行容器对象的 ACL:

```
dcecp> acl show /.:/hosts/silver/config/srvrexec
{user appl_admin criI}
{unauthenticated r}
{any_other r}
dcecp>
```

因为 **/.:/hosts/silver/config/srvrexec** 是容器，它还具有“初始容器 ACL”和“初始对象 ACL”。可以通过对 **acl** 操作使用 **-ic** 和 **-io** 选项来操作这些初始 ACL。注意，"初始容器 ACL“当前没有任何效果，因为子容器在 **/.:/hosts/***hostname***/config/srvrexec** 下不存在。

访问服务器执行信息的许可权： 每个服务器的配置信息都受她自己的 ACL 保护。这些 ACL 可以防止未授权的主体创建、更改或删除服务器配置信息，防止他们启动、停止、启用和禁用服务器。

每个 ACL 都是为它保护的服务器执行信息命名的，且具有类似下列的名称:

/.../*cellname*/**hosts/***hostname*/**config/srvrexec/***server_name*

ACL 具有下列许可权:

c （控制）
修改 ACL

r （读）
读服务器执行信息

w （写）
修改服务器执行信息

s （停止）
停止服务器。

例如，使用下列操作查看 ACL，获取主机 **silver** 上 **video-clip** 服务器的服务器执行信息:

```
dcecp> acl show /.:/hosts/silver/config/srvrexec/video_clip
{user appl_admin crws}
{unauthenticated r}
{any_other r}
dcecp>
```

此 ACL 从容器的“初始对象 ACL”获取缺省值。可以通过对 **acl** 操作使用 **-io** 选项，在“初始对象 ACL”上操作。下列示例显示 **video_clip** 服务器的“初始对象 ACL”:

```
dcecp> acl show /.:/hosts/silver/config/srvrexec/video_clip
-io
{unauthenticated r}
{any_other r}
dcecp>
```

配置服务器

使用 **server create** 操作，使应用程序服务器对服务器控制设施可访问。配置服务器意味着创建启动和控制服务器需要的信息。一般来说，这包括服务器启动命令行和变量，以及启动 DCE 应用程序需要的其它信息。

一些服务器在主机系统运行的任何时候都必须为可用。例如，可能希望提供主机活动信息的服务器在主机引导时启动，一直运行到主机关闭为止。可能需要其它类型的服务，或仅短期使用这些服务。服务器控制设施具有管理界面，能让您指定启动和停止服务器的一些条件:

- **显式**: 可以设置服务器以便在需要时显式启动它。
- **引导**: 可以设置服务器以在引导时启动。
- **自动**: 可以设置服务器按需求启动；即，当主机系统接收到客户机对服务的请求时，服务器将启动。
- **失败**: 可以设置服务器以在异常退出时自动启动。

下列示例为本地主机上虚构的视频剪切服务器 **video_clip** 创建了一个项。对于远程主机或另一个单元中的主机，请使用单元相关或全局名称。程序名 **vclip** 调用位于 **/usr/local/bin** 工作目录中的服务器。服务器具有编目方式，该方式通过将 **-catalog** 指定为变量来设置。服务器使用 DCE 安全性服务，因此服务器具有主体名称 **Vclip_Srv_1**。**-entryname** 选项在服务器存储绑定信息的单元目录服务 (CDS) 中指定项名。 **-starton** 选项设置服务器在 **dced** 接收到命名 **video_clip** 服务器的显式 **server start** 操作时启动。**failure** 属性进一步指定当服务器以失败状态退出时重新启动服务器。**-services** 选项具有注释信息，帮助管理员在此信息具有 **server show** 操作返回时标识服务器。需要 **interface** 属性，因为当服务器启动时 DCE 守护程序将此信息复制到主机端点映射表中。

```
dcecp> server create /.:/hosts/silver/config/srvrconf/video_clip
\
> -program {/usr/local/bin/vclip} \
> -directory {/tmp} -arguments {-catalog} \
> -principal {Vclip_Srv_1} \
> -entryname {/.:/subsys/applications/video_clip_1} \
> -starton {explicit failure} \
> -services {{annotation {Video Clip Catalog and Server}} \
> {interface {d860322b-d499-11cd-9dfb-0000c08adf56 1.0}}}
dcecp>
```

下一个示例配置相同的服务器，每当主机系统引导时启动。与前一个示例唯一的差异是 **-starton** 选项具有值 **boot**。

```
dcecp> server
create /.:/hosts/silver/config/srvrconf/video_clip \
> -program {/usr/local/bin/vclip} \
> -directory {/tmp} -arguments {-catalog} \
> -principal {Vclip_Srv_1} \
> -entryname {/.:/subsys/applications/video_clip_1} \
> -starton {boot} \
> -services {{annotation {Video Clip Catalog and Server}} \
> {interface {d860322b-d499-11cd-9dfb-0000c08adf56 1.0}}}
dcecp>
```

最后配置示例设置 **video_clip** 服务器，以当主机系统接收到客户机对服务的请求时启动。 **-starton** 选项值是 **auto**。第100页的『取消配置服务器』 讨论了禁用和启用服务的步骤。

```
dcecp> server create /.:/hosts/silver/config/srvrconf/video_clip \
> -program {/usr/local/bin/vclip} \
> -directory {/tmp} -arguments {-catalog} \
> -principal {Vclip_Srv_1} \
> -entryname {/.:/subsys/applications/video_clip_1} \
```

```
> -starton {auto} \
> -services {{annotation {Video Clip Catalog and Server}} \
> {interface {d860322b-d499-11cd-9dfb-0000c08adf56 1.0}}}
dcecp>
```

列出和检索服务器配置信息

当想在特定主机上配置服务器名称列表时，使用如下所示的 **server catalog** 操作。此操作不显示主机上可用的每个服务器，仅显示服务器配置数据库中存储的配置信息。

```
dcecp> server
catalog /.:/hosts/silver
/.../my_cell.goodco.com/hosts/silver/config/srvrconf/video_clip
dcecp>
```

通过使用 **foreach** 命令为单元中每个主机重复 **server catalog** 操作，在 DCE 单元中列出所有已配置服务器的名称。

```
foreach h [directory list /.:/hosts]{
   echo [server catalog $h]
}
```

如果不确定为服务器建立的配置信息，则可以使用如下所示的 **server show** 操作来查看它。使用 **-executing** 选项查看有关运行服务器的信息。

```
dcecp> server show /.:/hosts/silver/config/srvrconf/video_clip
{uuid d860322b-d499-11cd-9dfb-0000c08adf56 1.0}
{program {/usr/local/bin/vclip}}
{arguments {-catalog}}
{prerequisites {}}
{keytabs {683cf29a-e456-11cd-8f04-0000c08adf56}}
{services {{annotation "Video Clip Catalog and Server"}}
{principals {Vclip_Srv_1}}
{starton {explicit failure}}
{uid 1441}
{gid 1000}
{dir {/tmp}}
dcecp>
```

取消配置服务器

可以通过使用 **server delete** 操作从主机的配置数据库删除服务器配置信息。例如，当服务器移到不同的主机时，需要执行此操作。**server delete** 操作不停止当前正在运行的服务器。

下列示例从主机 **silver** 上的配置数据库删除 **video_clip** 服务器的配置信息:

```
dcecp> server delete /.:/hosts/silver/config/srvrconf/video_clip
dcecp>
```

启动和停止服务器

当服务器正确配置好以后，可以使用 **server start** 或 **server stop** 操作远程启动或停止服务器。例如，下列 **server start** 操作启动本地单元中主机 **silver** 上的服务器 **video_clip**:

```
dcecp> server start /.:/hosts/silver/config/srvrconf/video_clip
eb814e2a-0099-11ca-8678-02608c2ea96e
dcecp>
```

下一个示例停止本地单元中本地主机 **silver** 上的服务器 **video_clip**:

```
dcecp> server stop video_clip
dcecp>
```

禁用和启用服务

可以通过将客户机服务设置为禁用来防止客户机使用服务器提供的服务（即使服务器正在运行）。当设置为禁用时，服务器端点信息不会返回请求客户机，因此防止了客户机发现服务器。相反，客户机接收到端点未注册的服务器状态。但是，以前获取服务器端点的客户机仍能与服务器通信。

当服务器提供多个接口时，可以通过指定接口标识符来禁用一个或多个接口。下列示例禁用 **video_clip** 服务器的一个服务:

```
dcecp> server disable /.:/hosts/silver/config/srvrexec/video_clip \
> -interface {d860322b-d499-11cd-9dfb-0000c08adf56 1.0}
dcecp>
```

下一个示例在 **video_clip** 服务器的 **vidsrv** 服务被禁用后再次启用。此操作允许客户机获取服务器的端点。

```
dcecp> server enable /.:/hosts/silver/config/srvrexec/video_clip
\
> -interface {d860322b-d499-11cd-9dfb-0000c08adf56 1.0}
dcecp>
```

扩展服务器配置

一些服务器可能需要配置信息，与您的 DCE 软件一起提供的属性集不支持这些信息。可以通过使用 **xattrschema** 对象创建附加的*扩展注册表属性* (ERA) 以将任意信息添加到服务器配置信息中。

例如，假设您有一个服务器，需要指定对象系列的属性。您通过使用 **xattrschema** 对象创建这个属性。下列示例创建了称为 **srvrconf/objfamily** 的 ERA。该操作指定了查询、更新、测试和删除 ERA 需要的许可权，并指定了支持许可权的 ACL 管理。

```
dcecp> xattrschema create \
> /.:/hosts/silver/config/xattrschema/srvrconf/objfamily \
> -attribute {{annotation {object family}} {encoding uuid} \
> {aclmgr {srvrconf r w r d}}}
dcecp>
```

当创建了新的属性时，请使用第102页的『更改服务器配置』 中说明的 **servermodify** 操作来插入必需的数据。第327页的『第32章 创建与使用扩展注册表属性』 中提供了有关 ERA 的详细信息。

可以通过使用下列示例中显示的 **xattrschemashow** 操作来查看与 ERA 关联的属性:

```
dcecp> xattrschema show /.:/hosts/silver/config/xattrschema/srvrconf/objfamily
{aclmgr {srvrconf {{query r} {update w} {test r} {delete d}}}}
{annotation {object family}}
{applydefs no}
{encoding uuid}
{intercell reject}
{multivalued yes}
{reserved no}
{scope {}}
{trigbind {}}
{trigtype none}
{unique no}
{uuid 1bef2222-e687-11cd-b74a-0000c08adf56}
dcecp>
```

配置信息中的 ERA 受两种级别 ACL 保护。一种类型 ACL 保护 ERA 驻留的容器。第二种 ACL 类型保护个别 ERA。

ERA 容器 ACL 按下列方式命名:

/.../*cellname*/**hosts/***hostname*/**config/xattrschema**

ERA 容器 ACL 具有下列许可权:

c （控制）
修改 容器 ACL。

r （读）
读容器中的 ERA。

i （插入）
创建新的 ERA 信息。

l （插入）
虽然存在 **l** 许可权，它并不应用到 ERA 项。许可权应用到服务器控制设施，该设施在第95页的『访问服务器控制设施的许可权』中说明。

使用 **dcecp acl** 对象来查看或修改容器 ACL。例如，下列操作查看主机 **silver** 上的 ERA 容器 ACL:

```
dcecp> acl show /.:/hosts/silver/config/xattrschema
{user appl_admin criI}
{unauthenticated r}
{any_other r}
dcecp>
```

个别 ERA 的 ACL 按下列方式命名:

/.../*cellname*/**hosts/***hostname*/**config/xattrschema/***ERA_name*

个别 ERA 上的 ACL 可以防止未授权的主体创建、读取、更改或删除 ERA 信息。下列示例显示了为 **srvrconf/objfamily** ERA 建立的许可权。在此示例中，**c** 许可权无效果，因为它不是在使用 **xattrschema create** 操作创建 ERA 时指定的。所有的用户都能查询和测试 ERA。仅名称为 **appl_admin** 的用户同时还能更新和删除 ERA。

```
dcecp> acl show /.:/hosts/silver/config/xattrschema/srvrconf/objfamily
{user appl_admin crwd}
{unauthenticated cr}
{any_other cr}
dcecp>
```

此 ACL 从容器的“初始对象 ACL”获取缺省值。可以通过对 **acl** 操作使用 **-io** 选项，在“初始对象 ACL”上操作。下列示例显示主机 **silver** 上 **xattrschema** 容器的“初始对象 ACL”：

```
dcecp> acl show /.:/hosts/silver/config/xattrschema -io
{unauthenticated cr}
{any_other cr}
dcecp>
```

更改服务器配置

有时可能想更改服务器的配置信息。例如，想将 **-starton** 属性从 **boot** 更改为 **explicit**，以便能手工控制服务器。

要更改普通服务器配置属性，必须首先删除所有现有的属性，然后创建新的属性。可以通过首先使用 **server show** 操作在您的屏幕上显示以避免丢失当前信息。

详细步骤在下列示例中阐明，该示例使用 **server show** 操作捕捉当前服务器配置信息。**server delete** 操作删除配置信息， **server create** 操作插入新的 **-starton** 属性和剩余的服务器配置信息。

```
dcecp> server show /.:/hosts/silver/config/srvrconf/video_clip
{uuid d860322b-d499-11cd-9dfb-0000c08adf56 1.0}
{program {/usr/local/bin/vclip}}
{arguments {-catalog}}
{prerequisites {}}
{keytabs {683cf29a-e456-11cd-8f04-0000c08adf56}}
{services {{annotation "Video Clip Catalog and Server"}}
{principals {Vclip_Srv_1}}
{starton {boot}}
{uid 1441}
{gid 1000}
{dir {/tmp}}
dcecp> server delete /.:/hosts/silver/config/srvrconf/video_clip
dcecp> server create /.:/hosts/silver/config/srvrconf/video_clip \
> -program /usr/local/bin/vclip \
> -directory /tmp \
> -arguments {-catalog} \
> -principal Vclip_Srv_1 \
> -entryname /.:/subsys/applications/video_clip_1 \
> -starton {explicit} \
> -services {{annotation "Video Clip Catalog and Server"}}
dcecp>
```

可以通过使用 **server modify** 操作直接更改 ERA 信息。下列示例更改称为 **srvrconf/objfamily** 的服务器 ERA 以包含新值。此操作假设已经使用第101页的『扩展服务器配置』 中描述的 **xattrschema create** 操作创建了 ERA。

```
dcecp> server modify /.:/hosts/silver/config/srvrconf/video_clip
\
> -change {srvrconf/objfamily {c09dcc40-e4f4-11cd-bd59-0000c08adf56}}
dcecp>
```

检查服务器是否正在运行

可以通过执行 **server ping** 操作检查特定服务器是否正在运行。当一些客户机用户报告他们无法与服务器通信时，这可能是一个方便的测试。**server ping** 操作与名称服务器通信以测试它的存在，如果服务器正在侦听则返回 **1**，如果不在侦听则返回 **0**。server ping 操作的变量是服务器的 **entryname**，而不是 **srvrconf** 对象的名称。下列示例测试 **video_clip** 服务器是否正在运行:

```
dcecp> server ping /.:/subsys/applications/video_clip_1
1
dcecp>
```

管理客户机 / 服务器绑定信息

在 DCE 环境中，客户机和它们的服务器常常驻留在网络中不同的主机上，所以客户机需要一种方法来找到服务器。

客户机需要三个信息以与服务器通信:

- 其上正在运行服务器的主机的主机名（或网络地址）
- 服务器正在使用的网络传送名称

- 服务器用于客户机通信的通信端口（端点）

当然，应用程序员可以简单地将服务器的位置信息（也称为*绑定信息*）硬编码到可立即使用的应用程序的客户机方。但是，这种方法需要程序员具有精确网络细节的先进知识，例如主机名和可用的端口号。而且，具有硬编码绑定信息的服务器不会轻易适应配置更改。如果将服务器移动到不同的主机，则需要将所有客户机与服务器的新主机名称一起重新编译。所以，DCE 为客户机提供更多灵活的方式来获取服务器绑定。

客户机找到服务器的标准方式是使用 CDS 和服务器主机的端点映射表。图1提供这种方法的一个高级示例，显示主机 **larry** 上虚构的字典客户机应用程序如何找到主机 **curly** 上的字典服务器。

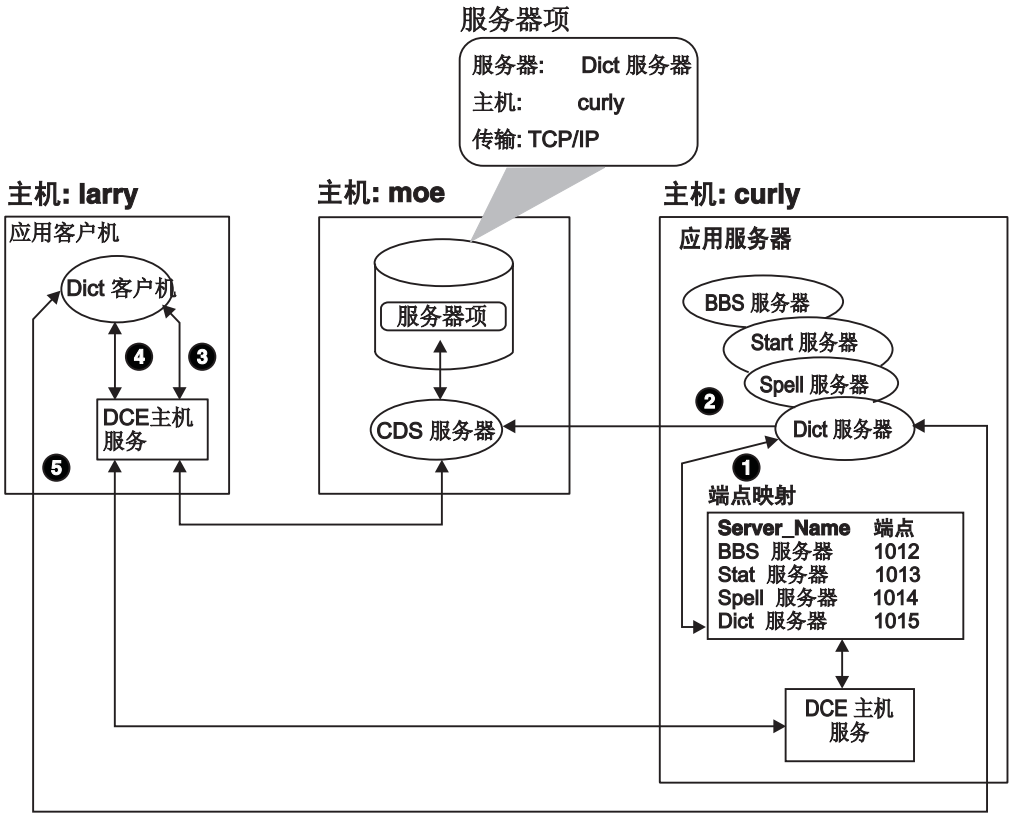


图 1. 服务器绑定信息

1. 当字典服务器启动时，DCE 主机软件指定服务器通信端口（端点），客户机将用它来与此服务器通信。这里，端点是 TCP/IP 端口 1015。DCE 主机软件将服务器标识信息与当前端点一起放入主机的端点映射表中。
2. 然后，字典服务器通过将它的主机名（通常是主机地址）和它使用的传送放入（*导出*） CDS 中的服务器项来向客户机介绍它的有效性。
3. 当字典客户机调用服务器提供的远程过程时，客户机上的 DCE 软件查询 CDS 服务器以找到字典服务器的主机名和传送。
4. 然后，客户机系统的主机软件查询主机 **curly** 上的端点映射表以查找字典服务器的端点（端口 1015）。
5. 配备了所有必需的绑定信息，主机 **larry** 上的主机服务将远程过程调用直接发送到主机 **curly** 上的端口 1015。

虽然我们在此高级示例中省略了一些详细信息，但图仍显示客户机和服务器执行的主要绑定活动。也就是说，服务器将它们的绑定信息放入 CDS 和客户机在其中寻找的主机端点映射表。客户机还可使用一些其它方法来查找服务器，这些方法在我们描述的机制上存在一些变异。但这些替代项通常是由应用程序本身，而不是通过类似 **dcecp** 的普通 DCE 管理设施控制的。

本节讨论一个基本的客户机／服务器绑定机制。下列部分检查端点映射表和 CDS 扮演的角色。还将讨论端点映射表和 CDS 中管理绑定信息的特定管理任务。

将端点映射表用于简易应用程序开发和管理

远程客户机可以通过使用服务器主机的端点映射表来查找服务器，以确定服务器的通信端点。但远程客户机如何才能知道到哪里去寻找端点映射表本身呢？它们知道，因为总能在每个主机上的*众所周知*端点访问端点映射表，客户机可以方便地找到它。

当主机支持多个传送时，端点将*侦听*映射到每个传送的一个端口上。在 IP 地址系列（TCP 和 UDP）中，端点映射表进程在端口 135 上侦听。在域域套接字 (DDS) 地址系列中，它在端口 12 上侦听。在 DECnet NSP 地址系列中，它在端口 69 上侦听。可以在头文件 **/opt/dcelocal/share/include/dce/ep.idl** 中找到端点映射表器服务使用的协议序列和众所周知端点的完整列表。

注意，不是所有的主机都支持所有的传送。DCE 软件尝试确保客户机和服务器之间共享至少一个传送。

当众所周知端点对一些关键服务器提供方便访问时，对于大多数服务器来说他们是不切实际的。这是因为一些地址系列具有有限的端点数，且众所周知端点仅能由中央管理机构指定。所以大多数服务器使用*动态端点*。当启动服务器时，RPC 运行时库从操作系统获取可用的端点，将它注册到主机端点映射表中。

因为服务器可以在每次启动时被指定为不同的端点，所以端点信息将存储在端点映射表中，而不是 CDS 中，CDS 是存放更稳定信息的资源库；这些信息是指服务器的主机地址和它使用的传送。只要服务器驻留在相同的机器上，主机和传送信息就不需要更新，这将减少 CDS 处的瓶颈。

这种方案使应用程序开发和管理更加简便，因为它减少了管理端点的需求。服务器需要关注将动态端点传送到客户机。而且，除非服务器移动到新的主机、删除或添加传送，它根本不需要更新 CDS 中的信息。

自动端点映射表管理

每个使用端点映射表的服务器都在启动时将一组信息存储到端点映射表中。信息包括服务器、注释字符串和其它字段为对象和接口提供的通用唯一标识符 (UUID)。

端点映射表驻留在 *dcelocal***/var/dced/Ep.db** 和 *dcelocal***/var/dced/Srvrexec.db** 中的磁盘上。在系统重新引导后，基于 DCE 的服务器使用端点映射表器服务重新启动和重新注册，因此在 DCE 守护程序启动之前需要删除数据库文件。这在大多数系统上都会自动发生。

基于 DCE 的服务器通常需要在启动时使用端点映射表器服务来注册，在终止时取消注册。如果任何服务器没有取消注册就退出，则端点映射表可能包含失效项。

DCE 提供服务器控制设施，帮助服务器取消注册并避免在端点映射表中留下失效项。不使用这些设施的服务器（例如，旧的服务器）如果异常退出，常会留下失效项。因此，DCE 守护程序 (**dced**) 定期地通过扫描端点映射表、ping 已注册的服务器，删除不响应的服务器项来清除失效项。

删除失效项的后台处理不是很高响应性的。当服务器不再服务 RPC 时，它不会替换服务器从端点映射表取消注册它们的需求。相反，这项处理仅是在服务器失败以后进行清理。

ping／清除机制的行为是从属实现的，在典型的实现中，仅偶尔扫描（指服务器 ping 和删除失效项）数据库；例如一小时几次。当 ping 服务器失败时，服务器将在短间隔内 ping 几次；例如，每 5 分钟。如果服务器继续不响应，**dced** 进程将确定它的项为失效的，将它从数据库中删除。最后，检测和清除失效服务器项的速率取决于数据库中的失效项数；失效项越多，检测和清除失效项的时间越长。

限制端点

可以限制 DCE 服务器和客户机的端点（端口）赋值到特定设置。如果您的环境具有与设计为使用某些端点的 DCE 不同的应用程序，且不想让 DCE 服务器或客户机独占它们，这种方法是很有用的。

此设施通过使用端点列表设置 **RPC_RESTRICTED_PORTS** 环境变量来激活，在启动客户机或服务器应用程序之前应该对这些端点列表限制动态赋值。**RPC_RESTRICTED_PORTS** 仅由 RPC 运行时管理动态服务器端点赋值。它不会影响众所周知端点。

下列示例限制服务器使用范围从 5000 至 5110 以及从 5500 至 5521 的 TCP/IP 端点。它限制 UDP/IP 端点到范围 6500 至 7000。

```
% set RPC_RESTRICTED_PORTS
\
        ncacn_ip_tcp[5000-5110,5500-5521]:ncadg_ip_udp[6500-7000]
%
```

要使用 DCE 服务器的 **RPC_RESTRICTED_PORTS**，例如 CDS，请在每次启动单元之前设置环境变量。

注意，此设施不会将任何安全性添加到 RPC，且不作为安全性功能使用。它仅帮助配置网络防火墙以允许对 DCE 服务器的进入调用。

查看端点映射表中的信息

对于大多数部件，每个主机上的端点映射表管理它本身，在必要时清除失效项，并在每次主机重新引导时删除端点信息。所以，端点映射表实际上不需要任何管理。

但是，当遇到客户机／服务器通信问题时，端点映射表中存储的信息可能对管理员有用，特别用于确定服务器是否提供正确的端点信息给客户机。在这种情况中，可以使用 **endpoint** 对象来查看端点映射表信息。除了用于故障检测，还可以为其它专门服务器操作使用 **endpoint** 对象，例如将新的对象 UUID 添加到现有的映射中。

端点不受 ACL 保护。这意味着可以运行 **dcecp** 的任何人都可以使用他们主机上的 **endpoint show** 操作来查看单元中任何其它主机上的端点信息。其它端点操作，例如

创建或删除端点，仅能由登录到本地主机上的用户执行。对端点信息的本地访问不需要其它特殊的特权，例如系统管理员或 root 特权。

可以通过使用 **endpoint show** 操作，查看主机端点映射表数据库中存储的信息。下列示例显示远程主机 **megazoid** 上 **video-clip** 服务器的端点映射表信息。省略 *hostname* 变量以在本地端点映射表上操作。

```
dcecp> endpoint show /.:/hosts/megazoid
\
> -interface {2fa417e8-bb4c-11cd-831b-0000c08adf56 1.0}
{{object 99ff4fb8-c042-11cd-91cd-0000c08adf56}
 {interface {2fa417e8-bb4c-11cd-831b-0000c08adf56 1.0}}
 {binding {ncacn_ip_tcp 130.105.1.227 1028}}
 {annotation {Text Development Utilities}}}
dcecp>
```

可以通过使用不带选项的 **endpoint show** 操作，查看端点映射表中的所有端点。

管理 CDS 中的服务器项、组和概要

端点映射表作为服务器目录在主机上运行。类似地，CDS 作为单元中的服务器目录。在本章第一部分给出了高级示例，表明应用程序如何使用 CDS 存储相对稳定的绑定信息，例如服务器名称、主机地址(n.)和可用服务器上的传送。在本节中，我们将显示如何使用 CDS 设施以有意义的方式组织服务器和其它分布式对象。

下列章节中讨论的许多操作都在受 ACL 保护防止未授权访问的 CDS 目录上操作。有关 ACL 和 CDS 的详细信息，请参阅第155页的『第16章 控制对 CDS 名称的访问』。

使用唯一服务器项名称来标识个别服务器和对象

我们知道服务器将它们的绑定信息存储到 CDS，客户机可以在其中找到它们。但到目前为止，我们仍将 CDS 视为黑盒子。如果 DCE 单元仅由几个服务器或对象和很少用户组成， CDS 可以简单地作为一个可由服务器和客户机访问的数据文件。查找对象的唯一名称不会造成大的问题。您甚至可以设计一些有效的方案，保护对象不被未授权使用。但 DCE 单元可以包含成百上千个对象。大单元可以包含需要便利和有效方式的许多类似或甚至等同的服务器以将它们的服务提供给客户机。

DCE CDS 通过提供服务器用于存储绑定信息的分层（树形结构）名称系统来响应这种需求。CDS 类似目录分层文件系统，存储名称和其它信息，不存储文件。可以在它的层次结构上建立，使目录名称与公司的组织结构相应。

服务器具有的 CDS 名称类似于 **/.:/admin/finance/payroll/check_writer**。当此 **check_writer** 服务器将它的服务器项名称导出到 CDS，CDS 将把它存储到名称为 **/.:/admin/finance/payroll** 的目录中。因此，客户机不会将此 **check_writer** 与另一个名为 **/.:/admin/finance/accts_payable/check_writer** 的 **check_writer** 混淆。这样，唯一的服务器项名称满足关键管理需求，提供一种方式访问并控制个别服务器。

本书 第121页的『第4部分 单元目录服务』 提供关于 CDS、结构和 CDS 名称使用的详细信息。对于当前目的，了解 CDS 目录名称如何和为什么帮助使可能等同的服务器项唯一已经足够了。

由于服务器本身经常管理从 CDS 导出和删除它们的名称和绑定信息，有时管理员需要手工添加、更改或删除绑定信息。例如，当服务器主机异常崩溃，脱机很长一段时

间，它的驻留服务器无法将它们的项名和绑定信息从 CDS 中删除。客户机可能在寻找这些不存在的服务器上浪费很多时间。DCE 控制程序提供 **rpcentry** 对象，可使用它们来管理 CDS 中的服务器项名和绑定信息。

在获得实际管理任务之前，让我们先检查服务器项精确地查看我们将要管理什么。图2显示了服务器项中的可能信息。

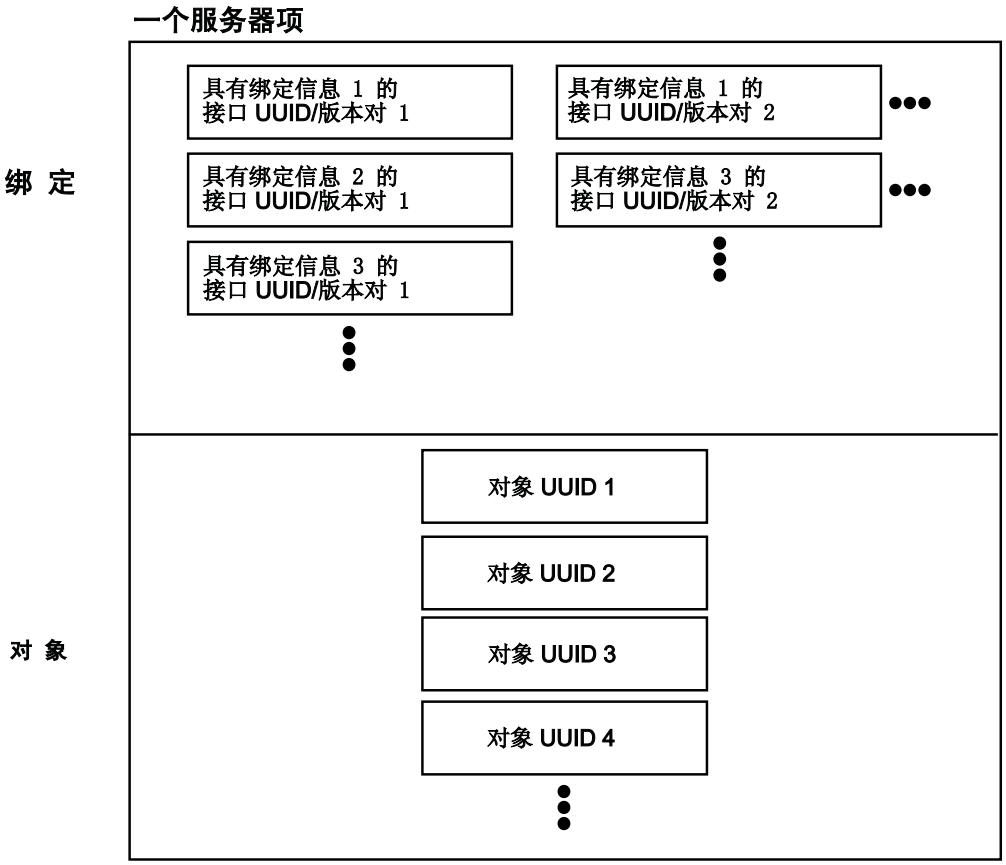


图 2. 服务器项中的可能信息

图2的顶部包含绑定。每个绑定都由接口标识符和绑定组成。接口标识符标识服务器提供的接口，绑定信息表明用于访问该接口的主机地址和网络传送。下列绑定示例（在 **dcecp** 语法中显示）表明服务器位于使用网际地址 120.101.13.157 的主机上，可以使用用户数据报协议 (UDP) 获取:

```
{nacdg_ip_udg 120.101.13.157}
```

当接口标识符在几个传送上可用时，服务器项包含绑定（每种传送一种绑定）。服务器可以提供多个接口。多个接口可以通过单一端点获取。即，不同的接口可以具有相同的绑定。

图的底部包含对象 UUID。对象 UUID 提供附加的信息给客户机；它们标识服务器管理的特定对象或资源。例如，一个打印服务器提供 2 楼的打印机，另一个打印服务器提供 1 楼的打印机。在这种情况中，对象 UUID 让客户机选择适当楼层上的打印机。换句话说，对象 UUID 帮助客户机从等同的服务中进行区别。

虽然应用程序服务器可以管理 CDS 中它们自己的服务器项，但您会发现手工添加、删除或更改服务器项中的信息会更加方便，更加简明易懂。有四种方法管理 CDS 中的服务器项:

- 可以将服务器项名硬编码到应用程序中。可以更改源代码中的服务器项信息，但在项名生效之前需要重新编译和重新运行应用程序。
- 可以将服务器项名存储为应用程序可以访问的服务器配置信息的 **entryname** 属性（使用 **server** 对象）。这比重新编译更方便，但更重要的是，这种方法将服务器项名放入管理员也能看到的标准（独立平台）位置。但是，可能需要重新启动应用程序来使用这种方法。
- 可以通过环境变量或变量将服务器项名传送到应用程序。虽然这些是有效的方法且比重新编译更方便，它们不是独立于平台的。这意味着在不同的操作系统上需要不同的方法。
- 可以使用 DCE 控制程序 **rpcentry** 对象在 CDS 中直接管理服务器项名。这种手工方法不需要重新编译或重新启动应用程序。

下一节讨论如何使用 **rpcentry** 对象在 CDS 中手工管理服务器项。

在 CDS 中创建服务器项

通常，当服务器初始化或当服务器在安装后配置时，将在 CDS 中创建它们自己的项。但有时，可能需要手工创建服务器项。当创建服务器项时，它是空的；不包含任何接口或绑定信息。

创建一个空的服务器项的原因是建立项的所有权。服务器项由创建程序拥有。如果服务器创建了一个项，它可以在稍后删除该项。可以通过创建项本身，抢先占有一个环境。稍后，服务器将它的绑定导出到现有的服务器项（倘若 ACL 允许的话）。

使用 **rpcentry create** 操作按下列方式创建空的服务器项，名为 **/.:/subsys/applications/bbs_server**。必须已经存在 CDS 目录 **/.:/subsys/applications** 以成功执行此操作。

```
dcecp> rpcentry create /.:/subsys/applications/bbs_server
dcecp>
```

从 CDS 删除服务器项

因为服务器项通常包含稳定的服务器绑定信息，它们会驻留其中，不会被删除。即使服务器脱机一段时间，例如一个晚上，仍不会实际删除它的项。但当服务器脱机很长一段时间时，可以通过从 CDS 删除服务器项来避免尝试使用不存在的服务器的客户机费用。

使用 **rpcentry delete** 操作从 CDS 删除服务器项，如以下所示:

```
dcecp> rpcentry delete
/.:/subsys/applications/bbs_server
dcecp>
```

将绑定信息导出到 CDS 中的服务器项

服务器通常在初始化或安装后配置时将自己的绑定信息导出到 CDS 中。但有时，会因为某些原因或意外删除绑定信息，可能想恢复它。或者，添加了另一个传送，想为新的传送导出绑定。

可以通过使用 **rpcentry export** 操作手工地将服务器绑定信息导出到服务器项。如果项不存在，**rpcentry export** 将假设目录已存在且您具有必要的许可权来创建它。

下列示例说明将服务器绑定信息导出到称为 **/.:/subsys/applications/bbs_server** 的服务器项。对象 UUID 标识 **bbs_server** 使用的数据文件资源。

```
dcecp> rpcentry export /.:/subsys/applications/bbs_server \
> -interface {458ffcbe-98c1-11cd-bd93-0000c08adf56 1.0} \
> -binding {ncacn_ip_tcp 130.105.1.227} \
> -object {76030c42-98d5-11cd-88bc-0000c08adf56}
dcecp>
```

从 CDS 中的服务器项导入绑定信息

应用程序客户程序可以自动从 CDS 导入服务器绑定信息，并在请求中使用它以查找并与服务器通信。但偶尔，管理员可能想导入联结点。例如，客户机可能对 CDS 缺乏访问权，但如果使用有效的绑定提供它则仍能与服务器通信。

使用 **rpcentry import** 操作返回服务器的绑定信息，如下所示:

```
dcecp> rpcentry import
/.:/subsys/applications/bbs_server \
> -interface {458ffcbe-98c1-11cd-bd93-0000c08adf56 1.0}
{ncacn_ip_tcp 130.105.1.227}
dcecp>
```

查看服务器项中的信息

当客户机与服务器通信有困难时，可能想知道什么绑定信息包含在服务器项中作为故障检测步骤。或想将对象 UUID 添加到服务器项，您不知道是否忽略了服务器项。可以使用 **rpcentry show** 操作查看下列示例中说明的服务器项中的信息。返回的信息包括接口标识符，服务器可以达到的两个绑定以及服务器维护的资源的对象 UUID。

```
dcecp> rpcentry show /.:/subsys/applications/bbs_server
{458ffcbe-98c1-11cd-bd93-0000c08adf56 1.0
  {ncadg_ip_udp 130.105.1.227}
  {ncacn_ip_tcp 130.105.1.227}}
{76030c42-98d5-11cd-88bc-0000c08adf56}
dcecp>
```

从 CDS 中的服务器项删除绑定信息

有时，想从服务器项处删除绑定信息。如果服务器主机崩溃了，它的服务器无法从 CDS 删除服务器项。要防止客户机尝试与这些不存在的服务器通信，应该手工从 CDS 中不导出绑定。与 **endpoint delete** 操作不同，此操作不从 CDS 中删除项名称。

使用 **rpcentry unexport** 操作按下列示例中显示的方法删除绑定信息。注意，不从服务器项删除对象 UUID 除非指定它为 **unexport** 操作的一个选项。

```
dcecp> rpcentry unexport /.:/subsys/applications/bbs_server \
> -interface {458ffcbe-98c1-11cd-bd93-0000c08adf56 1.0}
dcecp>
dcecp> rpcentry show /.:/subsys/applications/bbs_server
{76030c42-98d5-11cd-88bc-0000c08adf56}
dcecp>
```

使用组项来帮助平衡服务器工作负荷

当客户机查询服务器绑定的 CDS 时，请求包括项名称以查询绑定。当只有一个服务器提供客户机请求的服务时，CDS 将为请求此服务的每个客户机请求返回相同的绑定。此模型对有限客户机请求能很好地工作，但当许多客户机会聚在一个服务器上请求时可能会导致服务瓶颈。客户机通过提供多个服务器为较多数目的客户机请求服务来避免

瓶颈。服务器项名称本身不会提供便利的方式将客户机请求均匀分布到多个服务器上，因为必须明确地将每个客户机导向特定服务器。所以 CDS 提供*组项*作为便利机制将客户机负载分布到多个服务器上。

CDS 组项将相关的服务器集中到公共组名下。组项包含一般指向服务器项的成员，但成员也可以指向其它组项。当客户机任意从组项、CDS 返回请求绑定时，组项中将包含其中一个指针。如果任意选取的项是另一个组项，则 CDS 不会返回它。相反，CDS 达到该组，选取另一个随机成员，继续直到服务器项返回为止。此模型需要任何组成员能服务客户机请求。图3显示组项如何包含指向其它组和服务器项的成员。

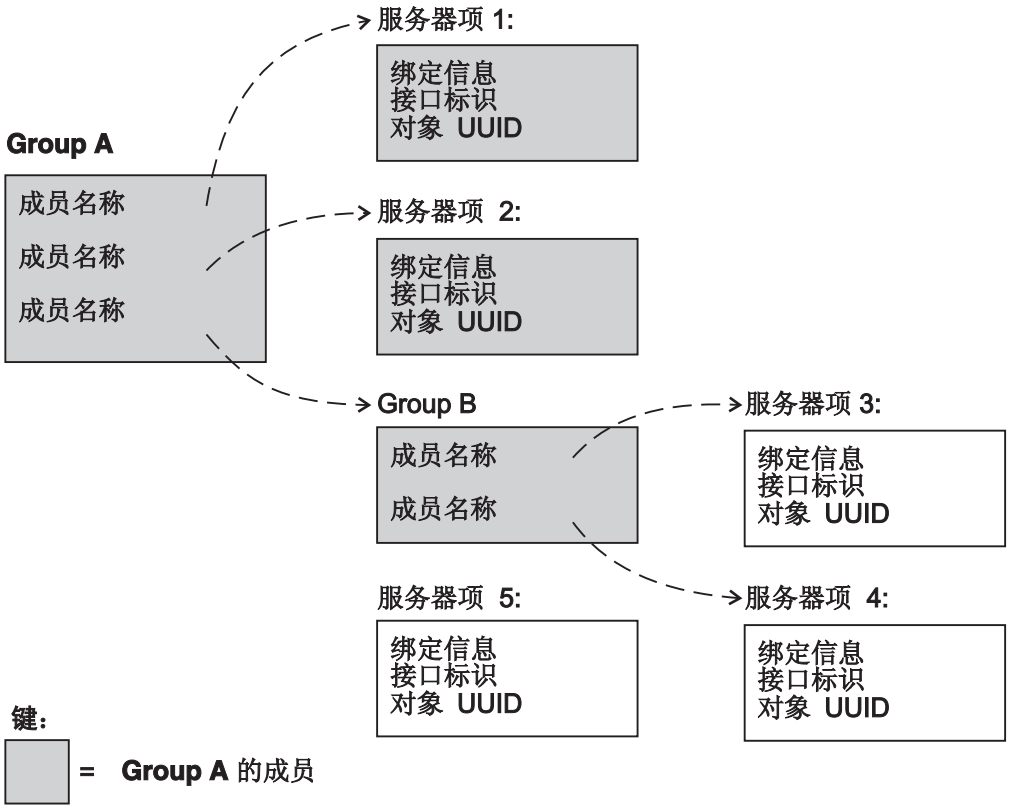


图 3. 组的可能映射

现在，让我们看一下组项是如何帮助平衡工作负荷的。考虑一个组织，具有 12 个相等激光打印机均匀分布在三个部门。下列组项示例显示每个组项名称如何返回赋值到它自己部门四台打印机的任何一个:

```
Group entry name: /.:/admin/finance/accts_payable_printers
  /.:/admin/finance/accts_payable/laser_10
  /.:/admin/finance/accts_payable/laser_11
  /.:/admin/finance/accts_payable/laser_12
  /.:/admin/finance/accts_payable/laser_13
Group entry name: /.:/admin/finance/accts_receivable_printers
  /.:/admin/finance/accts_receivable/laser_10
  /.:/admin/finance/accts_receivable/laser_11
  /.:/admin/finance/accts_receivable/laser_12
  /.:/admin/finance/accts_receivable/laser_13
Group entry name: /.:/admin/finance/payroll_printers
 /.:/admin/finance/payroll/laser_10
 /.:/admin/finance/payroll/laser_11
 /.:/admin/finance/payroll/laser_12
 /.:/admin/finance/payroll/laser_13
```

可以通过将组名添加到其它组的组项，暂时使一个部门的打印机被另一个组可用，如下一个组项示例所示:

```
Group entry name: /.:/admin/finance/accts_payable_printers
  /.:/admin/finance/accts_payable/laser_10
  /.:/admin/finance/accts_payable/laser_11
  /.:/admin/finance/accts_payable/laser_12
  /.:/admin/finance/accts_payable/laser_13
  /.:/admin/finance/accts_receivable_printers
```

前面示例中的配置表示应付帐中的客户机能使用应收帐户中打印机的 20%。可以通过添加服务器项名称而不是组名来提供较高使用百分率。下一个组项示例显示这样一种情况，应付帐户的客户机能使用应收帐户中打印机的 50%。但是，不要尝试通过多次包含组名来增加使用百分率，因为可能会发生错误。

```
Group entry name: /.:/admin/finance/accts_payable_printers
  /.:/admin/finance/accts_payable/laser_10
  /.:/admin/finance/accts_payable/laser_11
  /.:/admin/finance/accts_payable/laser_12
  /.:/admin/finance/accts_payable/laser_13
  /.:/admin/finance/accts_receivable/laser_10
  /.:/admin/finance/accts_receivable/laser_11
  /.:/admin/finance/accts_receivable/laser_12
  /.:/admin/finance/accts_receivable/laser_13
```

虽然应用程序服务器可以管理 CDS 中它们自己的组项，但您会发现手工添加、删除或更改组项中的服务器信息会更加方便，更加简明易懂。同管理服务器项类似，有多种在 CDS 中管理组项的方法:

- 可以将组项名硬编码到应用程序中。可以更改源代码中的组项信息，但在项名生效之前需要重新编译和重新运行应用程序。
- 可以通过环境变量或变量将组项传送到应用程序。这些是比重新编译更方便的方法，但可能需要重新启动一个应用程序来使用任何一种方法。
- 可以使用 DCE 控制程序 **rpcgroup** 对象在 CDS 中直接管理组项名。这种手工方法不需要重新编译或重新启动应用程序。

下一节讨论如何使用 **rpcgroup** 对象在 CDS 中手工管理组项。

在 CDS 中创建新的组项

可以通过使用 **rpcgroup create** 操作，在 CDS 中创建空组项。当首先使用组项的应用程序经常执行组创建时，创建项将使您成为项的所有者。作为所有者，对谁能导出和管理项中的信息有最终控制权。

要在 CDS 中创建空组项，请如下列示例中所示使用 **rpcgroup create** 操作。

```
dcecp> rpcgroup create
/.:/subsys/applications/admin_bbs_servers
dcecp>
```

将成员添加到 CDS 中的组项

可以使用 **rpcgroup add** 操作将成员添加到组项。如果组项不存在，操作将创建组项并添加成员。成员可以是服务器项或另一个组项。注意，没有任何操作会检查您实际添加的成员是否存在。这允许您在启动和运行服务器之前配置名称空间。

要将成员添加到 CDS 中的 **/.:/subsys/applications/admin_bbs_servers** 组项，请如下列示例所示使用 **rpcgroup add** 操作:

```
dcecp> rpcgroup add /.:/subsys/applications/admin_bbs_servers \
> -member /.:/subsys/applications/bbs_server4
dcecp>
```

查看组项的成员

可以通过使用 **rpcgroup list** 操作列出组项的成员。这对故障检测或仅对查看服务器如何在组项中分布很有用。

要在 CDS 中列出组项的成员，请使用 **rpcgroup list** 操作，如下列示例所示，列出组 **/.:/subsys/applications/admin_bbs_servers** 的成员:

```
dcecp> rpcgroup list /.:/subsys/applications/admin_bbs_servers
/.../my_cell.goodco.com/subsys/applications/bbs_server3
/.../my_cell.goodco.com/subsys/applications/bbs_server4
dcecp>
```

从 CDS 中的组项导入绑定信息

应用程序客户程序可以自动从 CDS 导入服务器绑定信息，并在请求中使用它以查找并与服务器通信。但偶尔，管理员可能想导入联结点。在某种情况中，客户机可能对 CDS 缺乏访问权，但如果使用有效的绑定提供它则仍能与服务器通信。

可以使用 **rpcgroup import** 操作返回服务器的绑定信息。必须使用 **-interface** 选项指定接口，如下列示例所示:

```
dcecp> rpcgroup
import /.:/subsys/applications/admin_bbs_servers \
> -interface {458ffcbe-98c1-11cd-88bc-0000c08adf56 1.0}
{ncacn_ip_tcp 130.105.1.227}
dcecp>
```

可以使用其它选项（例如 **-version** 和 **-object**）进一步指定绑定。使用 **-max** 选项限制返回的绑定数。

从 CDS 中的组项删除成员

随着时间的流逝，组织性更改可以要求您在 DCE 单元中重新部署服务器。例如，可能想将服务器项从一个组项移动到另一个。

使用 **rpcgroup remove** 操作从一个组中删除一个或多个成员。下列示例从组 **/.:/subsys/applications/admin_bbs_servers** 删除**bbs_server3**:

```
dcecp> rpcgroup remove /.:/subsys/applications/admin_bbs_servers \
> -member /.../my_cell.goodco.com/subsys/applications/bbs_server3
dcecp> rpcgroup list /.:/subsys/applications/admin_bbs_servers
/.../my_cell.goodco.com/subsys/applications/bbs_server4
/.../my_cell.goodco.com/subsys/applications/bbs_server5
/.../my_cell.goodco.com/subsys/applications/bbs_server6
dcecp>
```

从 CDS 删除组项

组织更改或服务器重新部署可能会造成一些组过时。当想从 CDS 删除组项时，使用 **rpcgroup delete** 操作。下列示例举例说明从 CDS 中删除称为 **/.:/subsys/admin/temporaries/wp_services** 的过时组项:

```
dcecp> rpcgroup delete /.:/subsys/admin/temporaries/wp_services
dcecp>
```

使用概要来指导服务器的客户机搜索

组项提供客户机来自多个可用服务的随机选项。虽然组项能帮助负载均衡和资源分配，它的随机性质抵制精细调整。而且，它不提供特定客户机使用的区分服务器优先次序的方式。

概要提供一种互补方式来组织服务器，因为可以区分概要成员的搜索优先次序。（这些称为以前 DCE 版本中的*元素*。）成员通过提供下列信息来标识服务器:

- 接口标识符

  此字段是概要的密钥。接口标识符包含接口 UUID 和接口版本号。

- 成员名

  下列任一种目录服务项的项名称:

  1. 提供所请求的 RPC 接口的服务器的服务器项
  2. 相应于所请求的 RPC 接口的组
  3. 概要

- 优先级值

  优先级值（0 是最高优先级；7 是最低优先级）是由概要成员创建者设计的，用于帮助确定搜索次序以在类似优先级成员中随机选择。

- 注释字符串

  注释字符串使您能标识概要成员的目的。注释可以是任何文本信息；例如，一个与接口标识符关联的接口名称或与组关联的服务或资源的描述。

  与接口标识符字段不同，注释字符串不是一个搜索键。

概要是灵活的；它们包含可以指向服务器项、组和其它概要的成员。概要还能包含称为*缺省概要成员*的特殊成员。此可选成员应该指向缺省概要，通常是可在组织中服务于大多数用户需求的综合备份概要。第115页的图4显示了概要的一些可能映射。

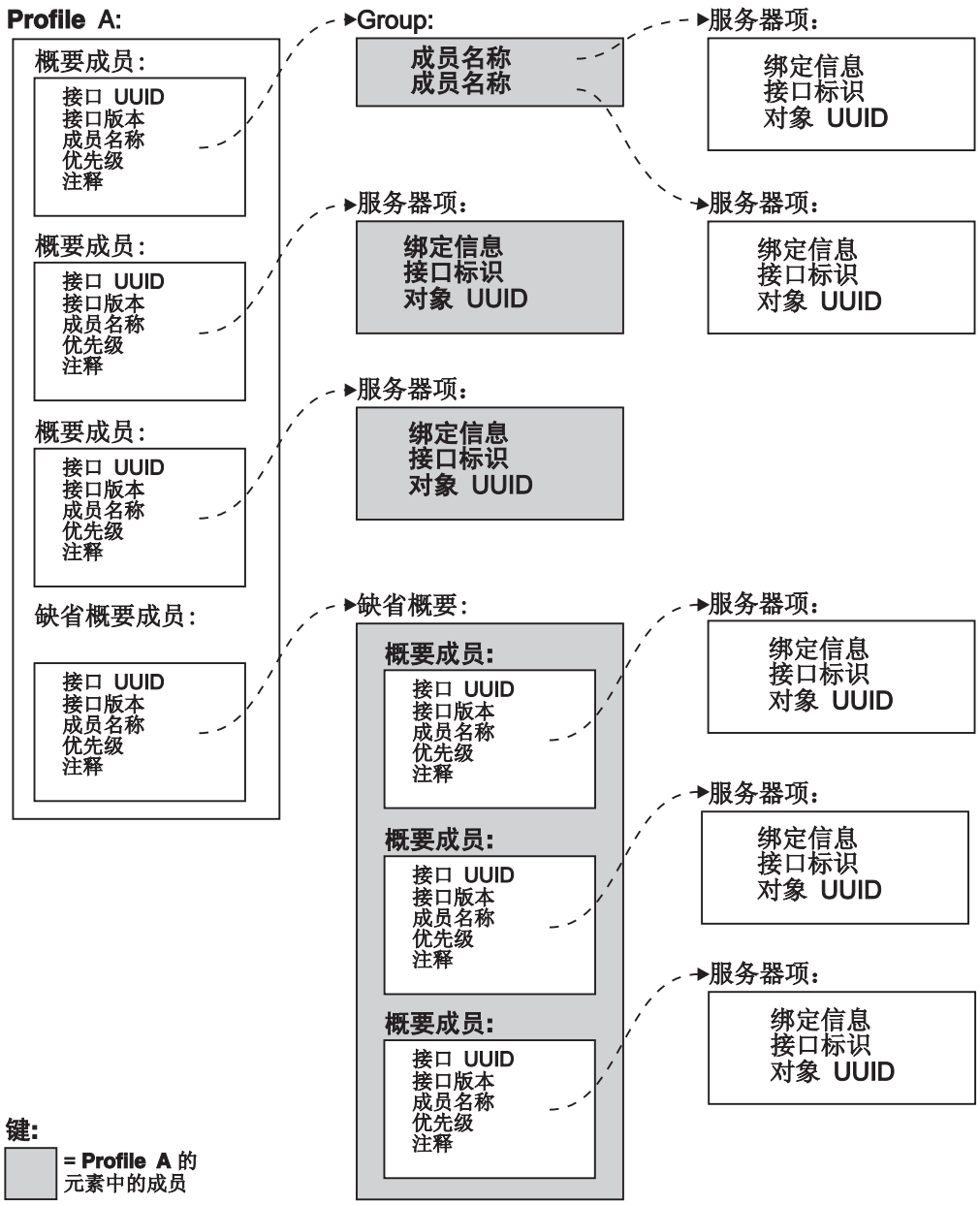


图 4. 概要的可能映射

要了解概要如何工作，让我们根据有关组项的以前讨论建立打印机示例。下列概要项示例显示了根据与客户机的亲近程度使用概要区分资源的优先次序的一种方法。

在图中，三个用户具有个人打印机概要，这些概要返回与它们最近的打印机的服务器项。例如，用户 **John** 最接近 **laser_20O**，所以概要优先级 **1** 首先返回该绑定。**John** 离 **laser_23** 最远，所以概要优先级 **4** 最后返回绑定。

```
Profile entry name: /.:/admin/finance/accts_receivable_printers/johns_profile
  /.:/admin/finance/accts_receivable/laser_20  1
  /.:/admin/finance/accts_receivable/laser_21  2
  /.:/admin/finance/accts_receivable/laser_22  3
  /.:/admin/finance/accts_receivable/laser_23  4
Profile entry name: /.:/admin/finance/accts_receivable_printers/pats_profile
  /.:/admin/finance/accts_receivable/laser_20  3
  /.:/admin/finance/accts_receivable/laser_21  4
```

```
  /.:/admin/finance/accts_receivable/laser_22  2
  /.:/admin/finance/accts_receivable/laser_23  1
Profile entry name: /.:/admin/finance/accts_receivable_printers/wills_profile
  /.:/admin/finance/accts_receivable/laser_20  2
  /.:/admin/finance/accts_receivable/laser_21  1
  /.:/admin/finance/accts_receivable/laser_22  3
  /.:/admin/finance/accts_receivable/laser_23  4
```

从这个示例得出推论，您的部门服务器被另一个部门过度使用了。可以通过在指向服务器的外来部门概要中降低服务器的优先级值，来进一步限制它的使用。

正如应用程序服务器可以在 CDS 中管理它们自己的概要项一样，它们还可以管理它们自己的概要项。但您会发现手工添加、删除或更改概要项中的服务器信息会更加方便，更加简明易懂。同管理服务器项和组项类似，有多种在 CDS 中管理概要项的方法:

- 可以将概要项名硬编码到应用程序中。可以更改源代码中的概要项信息，但在项名生效之前需要重新编译和重新运行应用程序。
- 可以通过环境变量或变量将概要项名传送到应用程序。这些是比重新编译更方便的方法，但可能需要重新启动一个应用程序来使用任何一种方法。
- 可以使用 DCE 控制程序 **rpcprofile** 对象在 CDS 中直接管理概要项名。这种手工方法不需要重新编译或重新启动应用程序。

下一节讨论如何使用 **rpcprofile** 对象在 CDS 中手工管理概要项。

创建新的概要

可以通过使用 **rpcprofile create** 操作，在 CDS 中创建空的概要项。当首先使用概要项的应用程序经常执行概要创建时，创建项将使您成为项的所有者。作为所有者，对谁能导出和管理项中的信息有最终控制权。

要在 CDS 中创建空概要项，请如下列示例所示使用 **rpcprofile create** 操作。

```
dcecp> rpcprofile create
/.:/subsys/applications/admin_group_profile
dcecp>
```

添加概要成员

可以使用 **rpcprofile add** 操作将成员添加到概要项。如果概要项不存在，操作将创建概要项并添加成员。成员可以是服务器项或另一个概要项。

要将成员添加到 CDS 中的 **/.:/subsys/applications/wards-profile** 概要项，请如下列示例所示使用 **rpcprofile add** 操作，使用优先级 **2** 添加服务器项 **/.:/subsys/applications/bbs_server3**:

```
dcecp> rpcprofile add /.:/subsys/applications/wards_profile \
> -member /.:/subsys/applications/bbs_server3 \
> -interface {458ffcbe-98c1-11cd-88bc-0000c08adf56 1.0} \
> -priority 2
dcecp>
```

查看概要项的成员

可以通过使用 **rpcprofile list** 操作简单列出概要项的成员。这对故障检测或仅对查看服务器如何在概要项中分布很有用。

要在 CDS 中列出概要项的成员，请使用 **rpcprofile list** 操作，如下列示例所示，列出概要 **/.:/subsys/applications/admin_bbs_servers** 的成员:

```
dcecp> rpcprofile list /.:/subsys/applications/wards_profile
/.../my_cell.goodco.com/subsys/applications/admin_bbs_servers
/.../my_cell.goodco.com/subsys/applications/bbs_server
dcecp>
```

可以通过使用 **rpcprofile show** 操作查看使用概要项存储的完整信息。这显示了与成员关联的优先级和接口 UUID。下列示例显示概要 **/.:/cell-profile** 中包含的所有信息:

```
dcecp> rpcprofile show /.:/cell-profile
{{d46113d0-a848-11cb-b863-08001e046aa5 2.0} /.../cell.co.com/sec 0 rs_bind}
{{0d7c1e50-113a-11ca-b71f-08001e01dc6c 1.0} /.../cell.co.com/sec-v1 0 secidmap}
{{8f73de50-768c-11ca-bffc-08001e039431 1.0} /.../cell.co.com/sec 0 krb5rpc}
{{b1e338f8-9533-11c9-a34a-08001e019c1e 1.0} /.../cell.co.com/sec 0 rpriv}
{{b1e338f8-9533-11c9-a34a-08001e019c1e 1.1} /.../cell.co.com/sec 0 rpriv}
{{6f264242-b9f8-11c9-ad31-08002b0dc035 1.0} /.../cell.co.com/lan-profile 0 LAN}
{{4d37f2dd-ed43-0000-02c0-37cf2e000001 4.0} /.../cell.co.com/fs 0 fs}
dcecp>
```

从 CDS 中的概要项导入绑定信息

应用程序客户程序可以自动从 CDS 导入服务器绑定信息，并在请求中使用它以查找并与服务器通信。但偶尔，管理员可能想导入联结点。在某种情况中，客户机可能对 CDS 缺乏访问权，但如果使用有效的绑定提供它则仍能与服务器通信。

可以使用 **rpcprofile import** 操作返回服务器的绑定信息。必须使用 **-interface** 选项指定接口，如下列示例所示:

```
dcecp> rpcprofile import /.:/subsys/applications/wards_profile \
> -interface {458ffcbe-98c1-11cd-88bc-0000c08adf56 1.0}
{ncacn_ip_tcp 130.105.1.202}
{ncacn_ip_tcp 130.105.1.227}
dcecp>
```

可以使用其它选项（例如 **-version** 和 **-object**）进一步指定绑定。使用 **-max** 选项限制返回的绑定数，如下列示例显示:

```
dcecp> rpcprofile import /.:/subsys/applications/wards_profile \
> -interface {458ffcbe-98c1-11cd-88bc-0000c08adf56 1.0} \
> -max 1
{ncacn_ip_tcp 130.105.1.202}
dcecp>
```

从 CDS 中的概要项删除成员

随着时间的流逝，组织性更改可以要求您在 DCE 单元中重新部署服务器。例如，可能想将服务器项从一个概要项移到另一个。

使用 **rpcprofile remove** 操作从一个概要中删除一个或多个成员。在下列示例中，**rpcprofile remove** 操作从概要 **/.:/subsys/applications/wards_profile** 删除成员 **/.:/subsys/applications/admin_bbs_servers**:

```
dcecp> rpcprofile remove /.:/subsys/applications/wards_profile \
> -member /.:/subsys/applications/admin_bbs_servers \
> -interface {458ffcbe-98c1-11cd-88bc-0000c08adf56 1.0}
dcecp>
```

从 CDS 删除概要项

组织更改或服务器重新部署可能会造成一些概要过时。当想从 CDS 删除概要项时，使用 **rpcprofile delete** 操作。下列示例举例说明从 CDS 中删除称为 **/.:/subsys/admin/temporaries/74232_profile** 的过时概要项:

```
dcecp> rpcprofile delete /.:/subsys/admin/temporaries/74232_profile
dcecp>
```

客户机管理

到目前为止，本章主要着重于服务器管理问题。我们已经知道如何控制一些服务器操作，如何将服务器绑定信息存储到 CDS 中和客户机在其中寻找的主机端点映射表中。本节讨论应用程序客户机的管理需求。虽然客户机管理很简单--只有两个相关操作--但它对使客户机和服务器共同工作来说是很关键的步骤。

我们知道 CDS 是一个目录分层系统，以服务器项格式存储服务器绑定信息。我们还知道 CDS 提供组项和概要项作为指导客户机到适当的服务器的一种方法。但客户机是如何知道到哪里去开始寻找服务器呢？

正如我们在本章前面章节中讨论的那样，服务器在 CDS 中注册接口和绑定。每个绑定接口的组合都在服务器项名下注册。当客户机执行远程过程调用时，它将服务器项名（或组项名或概要项名）与提供远程过程的接口的 UUID 一起传送到 CDS。CDS 将服务器项名（或组项名或概要项名）用作搜索绑定的起始点，该绑定包含与客户机传送匹配的接口 UUID 和版本。这种方法假设客户机以前已经获取了服务器使用的服务器项名（或组或概要名）。

让客户机使用适当的服务器项名是一个 2 步骤过程:

1. 确定客户机应该使用的项名。
2. 将名称传送给客户程序。

注意，客户机使用您提供的任何名称。客户程序无法区别名称是服务器项名、组项名还是概要项名。对客户机来说，所有这些名称的外观和行为都是相同的。

确定项名

需要知道服务器导出的项名，以便当配置它们时能将它提供给客户程序。在这里，我们称此名称为项名，但它可以是服务器项名、组项名或概要项名。应用程序文档应该帮助您决定使用哪种类型的项。

如果正在安装和配置应用程序的服务器和客户机部分，当配置服务器时请注意服务器的项名。

如果不在安装或配置服务器（例如，服务器是以前安装的），可能需要做一些检测工作来确定要使用的名称。可以在许多地方查询。

如果服务器使用本章中以前描述过的服务器控制设施，则可以使用 **server show** 操作来显示它的项名。当然，这意味着您需要知道服务器驻留的主机上的服务器对象名。可以使用 **server catalog** 操作查看主机上所有的服务器对象名。下列示例列出主机 **silver** 上配置的所有服务器对象。**server show** 操作显示 **info_server** 程序使用的项名。

```
dcecp> server catalog /.:/hosts/silver
/.../my_cell.goodco.com/hosts/silver/config/srvrconf/video_clip
/.../my_cell.goodco.com/hosts/silver/config/srvrconf/info_server
dcecp> server show /.:/hosts/silver/config/srvrconf/info_server
{uuid 6d5e7184-71b7-11cd-a205-08000925634b}
{program {/usr/local/bin/infosrv}}
{arguments {-brief}}
{prerequisites {}}
{keytabs {}}
{entryname {/.:/subsys/applications/info_server_1}}
{services {}}
```

```
{principals {}}
{starton {explicit failure}}
{uid 1423}
{gid 1000}
{dir {/tmp}}
dcecp>
```

如果服务器从某种类型的引导程序或脚本启动，则在程序或脚本中查看名称（有时，当服务器导出多个接口时，服务器使用多个名称）。可以以变量形式将名称提供给启动服务器的命令，如下列示例所示:

```
infosrv /.:/finance/operations/infoserv
```

当服务器方无法轻易显示它的项名时，尝试确定其它客户程序正在使用的项是什么。客户程序常常从某中类型的引导程序或脚本启动，项名一般以变量形式提供给启动客户机的命令。这些命令通常跟随在以前服务器启动命令示例中显示的相同模型之后。

将项名提供给客户机

有时，很简单的客户机可以在它们内编码服务器项名，所以您无需传送任何项名。但通常当客户程序启动时，需要将项名提供给客户程序。这种方式比硬编码项名更灵活，因为它提供了一种简单的方法在需求上升时使用不同的项名。

客户机配置文档应该包含如何将名称传送给客户机的指令。一种方法使用包含与变量（包含适当的服务器项名）一起启动客户机的命令的脚本或批处理文件。下列示例显示在启动客户机的外壳脚本中以命令变量传送的服务器项名:

```
# Shell Script to start the InfoClient application
infoclient /.:/finance/operations/InfoServ_profile
```

或者，可以将服务器项名存储在环境变量（在 UNIX系统上称为 **RPC_DEFAULT_ENTRY**）。下列示例显示定义此变量，然后调用客户机的外壳脚本:

```
#! /bin/sh
# Shell Script to start the InfoClient application
export RPC_DEFAULT_ENTRY=/.:/finance/operations/InfoServ_profile
infoclient
```

第4部分 单元目录服务

第11章 “DCE 目录服务”简介

分布式处理让多个系统进行交互来执行在传统计算环境中一个系统上执行的工作。这种网络范围的工作环境所带来的一个难题就是需要一种全球一致的方法来标识和定位网络中任何地方的人员与资源。

“DCE 目录服务”能够让人们在不知道他们物理位置的情况下进行联系，并使用网络中可能位于任何地方的资源，例如硬盘、打印队列和服务器。目录服务与电话目录辅助服务非常相似，后者只要给出人名就能提供电话号码。只要人员、服务器或资源的名称是唯一的，它就可以返回网络地址和其它与该名称相关的信息。

“DCE 目录服务”将地址和其它相关的信息作为名称的*属性*存储。例如，属性可以包含组织单位的名称，例如“欧洲销售部”；地址，例如 A 号楼一层；或者电话号码。

DCE 组件如何使用“DCE 目录服务”

“DCE 目录服务”是应用程序可以依赖和利用的一种基本服务。本节描述了其它 DCE 组件是如何使用“DCE 目录服务”的。

DCE 远程过程调用 (RPC) 接口利用了遵循客户机／服务器模型的分布式应用程序的开发和使用。在 RPC 模型中，*客户机*是进行远程过程调用的程序，而*服务器*是执行过程的程序。DCERPC 软件将有关 RPC 服务器地址和它们支持的接口的信息存储在目录服务中。

当 RPC 客户机希望对特定服务器进行调用时，它可以查询目录服务获得与该服务器联系所必需的信息。如果客户机希望访问在目录服务中命名的特定资源，它就可以查询该特定名称。如果客户机应用程序知道它所需要的服务的类型，例如 C 编译器、打印机或雇员信息，但不知道特定服务器的地址，它也可以使用目录服务来查找这些信息。

DCE 安全性服务在用户登录时验证他们的身份，并使用目录服务来存储认证服务器的地址。

分布式文件服务 (DFS) 提供了文件集（文件的逻辑组）的位置服务，使用户可以象文件在本地系统上那样访问远程文件。DFS 使用“DCE 目录服务”来了解如何与文件集位置服务器联系。

分布式时间服务 (DTS) 负责同步网络中的系统时钟。同步的时钟对于任何需要跟踪多个系统上事件发生顺序的分布式应用程序来说都很重要。DTS 使用“DCE 目录服务”来了解如何定位时间服务器。

如何使用“DCE 目录服务”

除 DCE 管理员以外，其他人通常都要通过一个应用程序接口来间接地使用目录服务。如果用户先为资源创建一个名称，然后根据名称来对资源进行引用，应用程序可以代表用户来与目录服务交互。下面的示例，无论是实际的还是假设的，都解释了一些用户对使用目录服务可以利用的方法:

- 某个用户对新文档调用拼写检查应用程序。该应用程序在用户的本地系统上包含了 DCE RPC 客户机代码。RPC 客户机与目录服务联系以查找有关可用拼写检查服务器的信息。目录服务返回服务器地址、通信的协议类型，以及表示接口的全局唯一标

识符 (UUID)。使用这些信息，RPC 客户机对服务器进行远程调用，然后服务器检查用户文档的拼写。用户不知道拼写检查程序的使用牵涉到了对目录服务的调用，也不知道与远程服务器的接口。

- 登录到某一系统的用户输入了名称和口令。目录服务帮助登录程序找到认证服务器，该服务器在认证数据库中验证用户的身份。
- 用户输入一个文件说明。目录服务提供 DFS 文件集位置数据库的地址，该地址包含让用户访问文件的服务器的网络地址。
- 用户输入计算机会议或电子公告板的名称，目录服务提供一个地址，让应用程序连接到会议服务。
- 通过输入名称或有关打印机能力的某些信息，用户可以知道打印机的网络地址。例如，用户可能希望查找最近和最快的可用彩色打印机的地址。
- 用户需要了解市场营销部门中某位职员的信息。该用户记得职员的姓是“王”，但不记得名。通过在职员定位应用程序中输入姓和部门名，该用户可以检查目录服务来知道市场营销部门中所有姓王的职员的信息，并知道如何与该职员联系。
- 用户在问题跟踪报告数据库中输入一份报告。尽管数据库最近转移到新节点上，但因为数据库总是按照其名称来引用的，所以用户并不知道这一变更。目录服务存储当前的网络地址，并把它提供给问题跟踪应用程序和所有其它请求它的应用程序。

本章的以下部分将解释“DCE 目录服务”环境是如何在单元中工作的。它介绍了一些主要的目录服务组件：单元目录服务 (CDS) 和全局目录代理 (GDA)，后者是本地和全局命名环境的网关。本章还讨论了域名系统 (DNS) 和 LDAP 服务器的 DCE 支持，它们是一些全局名称服务，但不作为 DCE 技术的一部分提供。

目录服务和单元环境

本节介绍了 DCE 命名环境的以下主要组件，并解释了它们与单元的关系：

- CDS
- DNS
- GDS
- LDAP 客户机／服务器
- GDA

CDS 是一种高性能的分布式服务，它为命名和使用单元内部（单元内）的资源提供了一致的、与位置无关的方法。CDS 也可以用于单元之间（单元间）的通信。

DNS 是一种广泛使用的现有全局名称服务，DCE 提供了对它的支持。当前有许多网络都主要使用 DNS 作为因特网主机名的名称服务。尽管 DNS 不是 DCE 技术提供的一部分，目录服务还是包含了单元通过 DNS 进行互操作所需的支持。 GDS X.500 服务器接受从 X.500 客户机通过目录访问协议 (DAP) 来访问其目录中的对象。在 DCE 中，服务器是 GDS 服务器，客户机是 GDS 客户机。GDA 通过 XDS/XOM API 与 GDS 客户机进行通信。GDS 客户机和服务器都基于 1988 X.500 标准。全局目录服务功能可以在 AIX 版本 3.2.5 中作为一个单独的产品使用。它在 AIX 版本 4.1 和更高版本中不受支持。

LDAP 客户机基于密歇根大学 3.3 源代码。 LDAP 客户机接受来自 GDA 的 LDAP API，并通过 LDAP 协议与 LDAP 服务器进行通信。

LDAP 服务器是一种服务器，它接受 LDAP 客户机的 LDAP 协议来访问其目录中的对象。LDAP 服务器可以是同样接受 LDAP 协议的 X.500 服务器或接受 LDAP 协议的任何专有目录服务。LDAP 服务器不由 DCE 提供，必须由用户提供。GDA 通过 LDAP API 与 LDAP 客户机进行通信。

图5 表示了一个具有两个单元的假设配置，每一个都使用 X.500 或 LDAP 服务器来访问另一个单元中的名称。也可以从每个单元访问直接存储在 X.500 或 LDAP 服务器中的名称。CDS 是每个单元中的目录服务。两个单元由同一个组织管理，它们是基于地理位置和网络拓扑进行的配置。

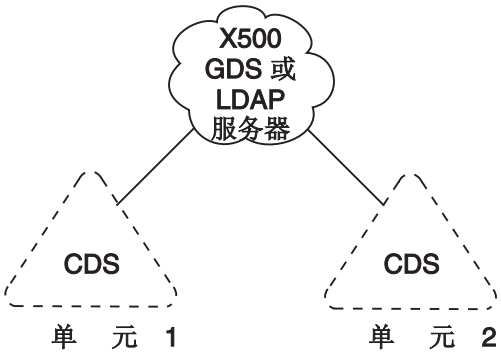


图 5. 单元和全局命名环境

GDA 是一种使单元互操作成为可能的 DCE 组件。GDA 使 CDS 能够通过一种全局命名环境（X.500、LDAP 或 DNS）访问另一个单元中的名称。GDA 是一个独立的进程，可以和 CDS 服务器存在于不同的系统上。CDS 需要能够与至少一个 GDA 联系以参与到全局命名环境中。

第126页的图6 显示了 GDA 如何帮助 CDS 访问单元外部的名称。当 CDS 确定某个名称不在它自己的单元内以后，它就将名称传递给 GDA，GDA 再搜索合适的命名环境(CDS、X.500、LDAP 或 DNS）来查找有关名称的详细信息。GDA 返回的信息能让原始 CDS 服务器与名称所在的单元中的 CDS 服务器联系。GDA 可以帮助 CDS 查找在 DNS 中登记的单元中的名称（方案 A）和在 X.500 或 LDAP 服务器中登记的单元中的名称（方案 B）。 GDA 根据名称的语法来确定使用哪个名称服务。第128页的『DCE 名称的深入分析』 详细描述了名称语法。请注意，GDA 和 X.500、GDA 或 LDAP 服务器之间的接口依赖于所使用的服务器的类型。GDA 使用 XDS/XOM API 作为与 GDS 客户机的接口。GDS 客户机使用 LDAP 协议作为与 X.500 服务器的接口。GDA 使用 LDAP API 作为与 LDAP 客户机的接口。LDAP 客户机使用 LDAP 协议作为与 LDAP 服务器的接口。

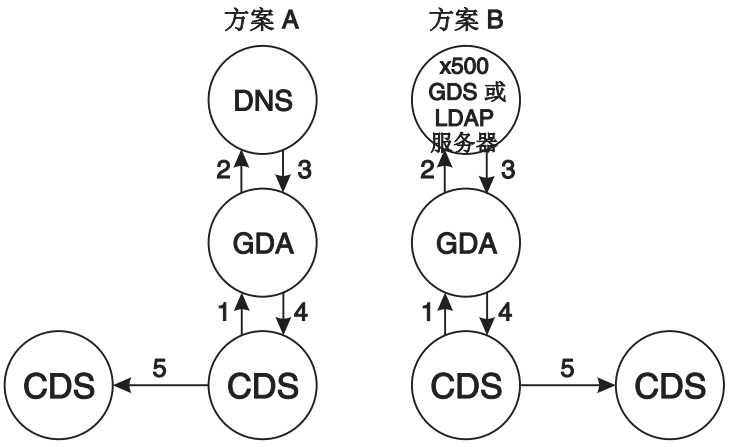


图 6. CDS、GDA 和全局目录服务的交互

单元如何确定命名环境

除了为用户和资源描绘安全性和管理界限之外，单元还为一组名称确定边界。因为不同的命名组件在单元内和单元外操作，所以单元中和全局环境中的命名约定也有所不同。DCE 命名环境支持两种名称: *全局名*和*与单元相关*的或者*本地*名。以下各小节介绍了全局名和本地名的概念。第128页的『DCE 名称的深入分析』 详细描述了 CDS 和 DNS 名称。

全局名

“DCE 目录服务”中的所有项都有一个在全球有意义的全局名，它可以从 DCE 命名环境中的任何地方使用。前缀 **/...** 表明名称是全局的。全局名可以指单元中的一个名称（以 CDS 命名），或单元外的一个对象（以 X.500 命名）。

下例显示了以 X.500 创建的某一项的全局名。该名称代表了用户 Ellie Bloggs，他在组织 Widget（一家英国公司）中的行政组织单位中工作。

```
/.../C=GB/O=Widget/OU=Admin/CN=Ellie Bloggs
```

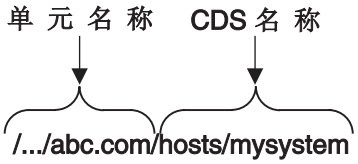
X.500 名称语法由一个全局前缀 **/...** 和一系列元素组成，称为相对专有名 (RDN)。每个 RDN 都包括一对或多对由 **=**（等号）字符分隔的部分。由等号分隔的项是多属性值断言 (AVA)。一对的第一部分是表明信息类型的缩写。某些常见缩写包括国家 (**C**)、组织 (**O**)、组织单位 (**OU**) 和常用名 (**CN**)。一对的第二部分是值。（请参阅第128页的『X.500 名称』 了解有关 X.500 名称的详细信息。）

下例显示了以 CDS 命名的价格数据库服务器的全局名。该服务器由 XYZ 公司（一家美国组织）的波特兰销售部门使用。

如示例显示，以 CDS 创建的项的全局名看上去与纯 X.500 样式的名称略有不同。名称的第一部分，**/.../C=US/O=XYZ/OU=Portland** 是在 X.500 服务器中存在的全局单元名。其余部分，**/subsys/PriceMax/price_server1**，是 CDS 名。

有这样的单元名存在是因为单元必须让名称能够在全局命名环境中访问到。GDA 在帮助一个单元中的 CDS 查找另一个单元中名称的进程中查找单元名。单元名在 DCE 组件的初始配置期间建立。在配置一个将参与到标准单元间通信的单元之前（即通过 X.500 全局目录服务），DCE 管理员必须依赖于单元需要通过 X.500 访问还是 DNS 访问来从两个全局命名环境之一获得唯一单元名。

下一个示例显示了 ABC 公司主机的全局名。公司单元的全局名 **/.../abc.com** 在 DNS 中存在。



独立单元中与单元相关的命名

除了它们的全局名以外，所有 CDS 项还都有与单元相关的，或本地名，它只在项存在的本地单元的内部有意义并使用。本地名是全局名的缩短形式，因此是在用户自己单元中引用资源的一种更方便的方式。本地名有以下特征:

- 它们不包括全局单元名。
- 它们以 **/.:** 前缀开始。

本地名不包括全局单元名，因为 **/.:** 前缀表明要引用的名称是在本地单元的内部。当 CDS 在名称中遇到一个 **/.:** 前缀时，它自动使用本地单元名替换前缀，组成全局名。CDS 可以处理全局名和本地名，但在引用本地单元中的名称时，使用本地名更方便一些。例如，这些名称在名为 **/.../C=US/O=XYZ/OU=Portland** 的单元中使用时是一样有效的:

```
/.../C=US/O=XYZ/OU=Portland/subsys/PriceMax/price_server1
/.:/subsys/PriceMax/price_server1
```

本地和全局目录服务交互所需要的命名约定最初时似乎很让人迷惑。在必须对本地单元外部名称进行引用的环境中，使用下面一些简单的指示可以有助于约定更容易记住和使用:

- 知道单元名。
- 知道您所引用的名称是否在单元内。
- 使用单元中的名称时，可以忽略单元名并包括 **/.:** 前缀。
- 使用单元外部的名称时，输入它的全局语法，包括 **/...** 前缀和单元名。
- 当有人问起单元中资源的名称时，提供它的全局名，包括 **/...** 前缀。
- 当在持久存储器（例如外壳脚本）中存储名称时，使用它的全局名，包括 **/...** 前缀。本地名（即带有 **/.:** 前缀的名称）只用于交互式使用，不应该存储它。（如果本地名是从外部单元中进行引用的，**/.:** 前缀将解析成外部单元的名称，名称查表的结果或者失败，或者产生错误的名称。）

DCE 名称的深入分析

本章其余部分提供了组成 DCE 名称空间不同种类名称的深入描述。附录 A 包含了 CDS 和 DNS 名称中有关有效字符和命名约定的进一步详细信息。

CDS 名称

每个单元都至少包含一个运行 CDS 服务器的服务器。 CDS 服务器存储和维护名称，并处理创建、修改和查找数据的请求。在单元中由 CDS 服务器共享的总的名称集合称为*单元名称空间*。单元名称空间管理员可以将 CDS 组成目录的不同层次结构。 CDS 目录在概念上与操作系统文件系统中的目录类似，它们都是将名称进行分组以简化管理和使用的逻辑方式。

在单元名称空间中，任何其下具有目录的目录被认为是其下面目录的*父代*。任何其上具有目录的目录被认为是其上目录的*子代*。单元名称空间的最顶层称为*单元根*。可以通过单元的全局名或简短格式 **/.:** 前缀来引用单元根。

图7 显示了从单元根开始的一个简单的单元名称空间层次结构。单元根 (**/.:**) 是名为 **/.:/hosts** 和 **/.:/subsys** 单元的父代。**/.:/subsys** 目录是单元根目录的子代，**/.:/subsys/dce** 目录的父代。

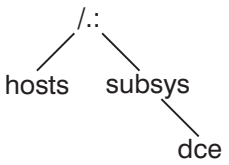


图 7. 样本 CDS 名称空间层次结构

CDS 名称的完整规范，从左至右，从单元根到指定的项，称为*全名*。全名中的每个元素都由 /（斜杠）分隔，称为*简名*。例如，假设 图7 中显示的 **/.:/hosts** 目录包含简名为 **bargle** 的主机项。该项的 CDS 全名就是 **/.:/hosts/bargle**。多个连续的斜杠将转换为全名中单一斜杠。

多个目录层次可以在分发、控制访问，以及管理许多名称时具有灵活性。目录层次结构还减少了重复名称的可能性。例如，名称 **/.:/subsys/Hypermax/printQ/server1** 和 **/.:/subsys/ABC/spell/server1** 都是唯一的。

X.500 名称

X.500 的操作与 CDS 的操作类似，但在名称结构上和查找方式上有些重大的不同之处。与 CDS 类似，X.500 和 LDAP 服务器都具有提供对 X.500 名称访问和管理的服务器进程。该进程称为目录系统代理 (DSA)。参与到同一个全局目录服务实现的所有 DSA 组合起来的信息称为目录信息库 (DIB)。该集合体信息被视为由许多项组成的单一全局目录。

X.500 全局目录中以根层次结构存在的信息称为目录信息树 (DIT)。DIT 与 CDS 名称空间类似。不过与名称空间（它没有有关结构和内容的继承规则）不同，X.500 层次结构受一系列称为*模式*的规则影响。每个 X.500 DSA 都必须定义一个标准模式，它的 DIB 部分中的所有项都要遵守这一模式。

尽管 X.500 标准不强制使用特定的模式，但它确实为电子邮件制定了常规建议书，这些建议书在很大程度上基于现有的 X.400 标准。例如，国家和组织的命名应该接近 DIT 根；人员、应用程序和设备的命名应该在层次结构中比较远的地方。X.500 提供了符合这些建议书的缺省模式。

每个 X.500 项都有一个专有名称，它唯一而明确地标识该项。专有名称由一系列有效的相对专有名称 (RDN) 组成。每个 RDN 在 DIT 的特定部分都包括一个或多个属性类型和值的断言。属性类型表明存储在属性值中的信息的性质。由属性类型和属性值组成的一对又称为属性值断言 (AVA)。RDN 可以有多个 AVA。例如，专有名称:

```
/C=us/O=osf/OU=branch1/CN=nollman,OU=doc-team
```

由四个 RDN 组成。最后一个 RDN 包括了两个由逗号分隔的 AVA。

图8 说明了 RDN 和专有名称的概念以及它们是如何与 DIT 相关的。该图显示了以下内容:

- DIT，由模式定义的属性类型的层次结构组成
- RDN，由属性类型和值的断言产生
- 专有名称，由 RDN 并置产生

X.500 名称由 GDA 理解，它通过 XDS/XOM API 与 X.500 客户机 (GDS) 联系，或者通过 LDAP API 与 LDAP 客户机联系以解析 X.500 单元名。

由 LDAP 客户机联系的 LDAP 服务器可以是专有的，也可以是支持 LDAP 访问协议的 X.500 服务器。因此，需要与 LDAP 服务器的供应商联系以获得这些信息。

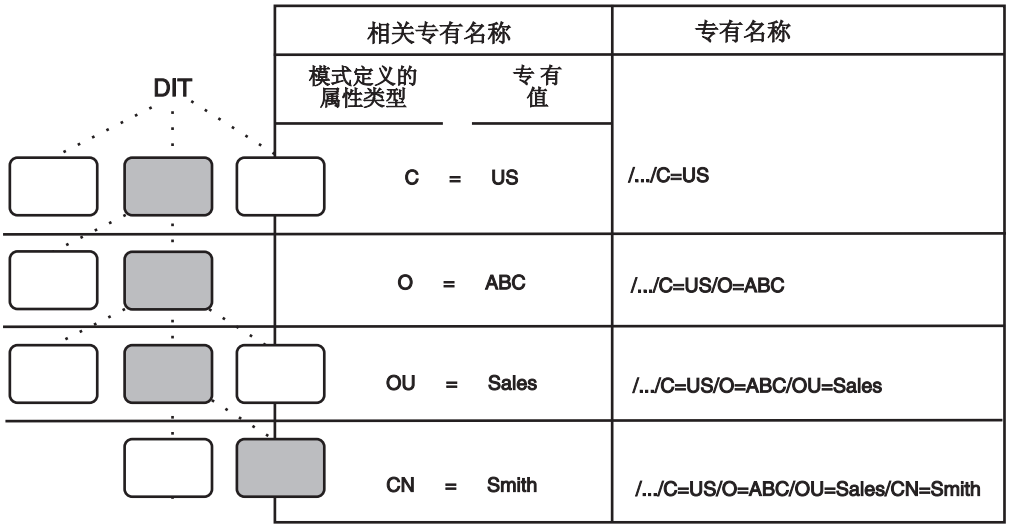


图 8. RDN 和专有名称

DIT 中带有阴影的框表示在标签为相对专有名称列中命名的项。该模式规定国家直接在根下命名，后面跟组织、组织单位以及用户名。每个组成 RDN（因此组成专有名称）的属性值都称为*专有值*。

如图中最右边的一列显示，DIT 每个级别的项的专有名称都是从全局目录根到该项级别的 RDN 的并置。层次结构中最低的那一项 `/.../C=US/O=ABC/OU=Sales/CN=Smith` 表

示的是一个用户名 John Smith，他在 ABC 公司（一家美国组织）的销售部门工作。缩写的属性类型标签代表国家 (C)、组织 (O)、组织单位 (OU) 和常用名 (CN)。

请注意，图中显示了专有名称的全局 DCE 约定。每个专有名称都以全局根 (/...) 的表示法开始。属性类型和值由等号分隔，RDN 由斜杠分隔。不是所有 X.500 实现都遵循指定名称的这些约定。另外，这些约定只在 X.500 管理界面级别上使用。在内部，专有名称以另一种方式指定。

X.500 名称的结构指出 X.500 和 CDS 之间的另一项重要差异。CDS 名与其属性区分开；即，它包含了以项的简名结尾的目录名字符串。与之相反，X.500 名称只包含一系列属性和它们的值。

图9 说明了在 CDS 和 X.500 名称构造中的这一差异。CDS 全名 /.:/Admin/Personnel/Employee_DB 是带有简名 Employee_DB 的项的完整目录规范。属性和它们的值不是 CDS 全名的一部分。X.500 专有名称 /.../C=US/O=ABC/OU=Sales 是属性类型和值的并置，它们每个都来自 DIT 模式的每一层。

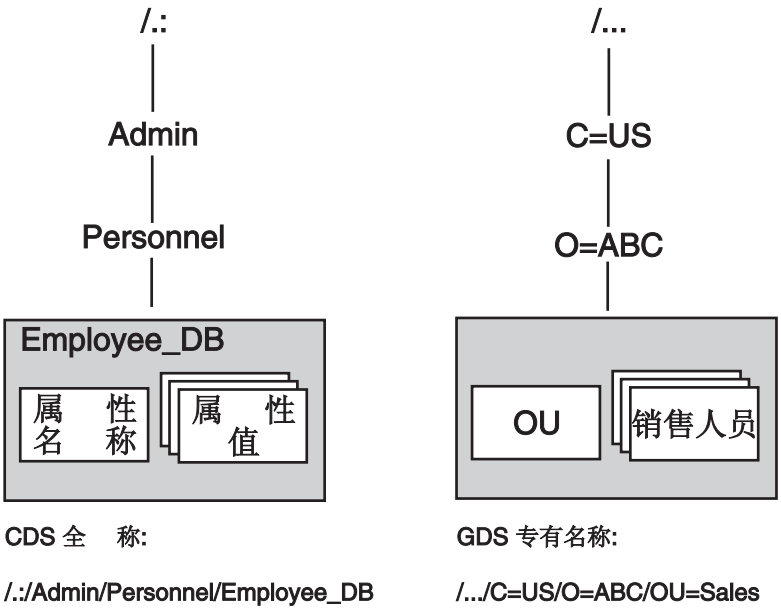


图 9. CDS 和 X.500 名称比较

注: LDAP 名称 /.../OU=Sales,O=ABC,C=US 在 DCE 中无效。名称必须按照 X.500 专有名称 (/.../C=US/O=ABC/OU=Sales) 指定。

X.500 支持通过提供一个或多个属性值来搜索名称的能力。这产生一种称为*描述性命名*的方法；在某种意义上，用户可以描述他们所查找的名称。尽管搜索能力很有价值，但它也是昂贵而耗时的；因此 X.500 允许用户限制搜索范围。搜索操作支持只限于 X.500 环境。

LDAP 名称

LDAP 名称包含与 X.500 名称相同的信息，但在语法上有所不同。LDAP 名称以 X.500 名称的最后一个 RDN 开始，使用逗号(,) 而不是斜杠 (/) 作为 RDN 分隔符。下例显示了这些差异:

```
X.500 名称: /C=us/O=osf/OU=branch1/CN=nollman/OU=doc_team
LDAP 名称: OU=doc_team,CN=nollman,OU=branch1,O=osf,C=us
```

DCE 只支持 X.500 单元名。GDA 在通过 LDAP 客户机访问 LDAP 服务器时会将 X.500 单元名转换成 LDAP 语法。

DNS 名称

DCE 命名环境支持基于因特网请求建议书 (RFC) 1034 和 RFC 1035 的 DCE 版本。当前有许多网络主要使用 DNS 作为主机名名称服务。最常用的 DNS 实现是 Berkeley 因特网命名域 (BIND)。BIND 名称空间是一种分层的树，它的最顶层在网络信息中心 (NIC) 的控制下。（请参阅 *IBM DCE 版本 3.2 AIX 和 Solaris 版: 管理指南--简介* 了解有关如何与 NIC 域注册部门联系来注册域名的信息。）

BIND 名称空间根下面紧接着的名称包括用于国家的 2 字母的代码，例如 **us** 和 **gb**，这些代码在 ISO 标准 3166，『国家名表示代码』中定义。根下面的其它名称包括一些常规管理类别，例如 **com**（商业）、**edu**（教育）、**gov**（政府）和 **org**（其它组织）。这些名称的所有者可以对公司和组织授予创建新的附属名称的许可权。图10 显示了 BIND 名称空间的一个样本部分。（双引号表明名称空间的根具有空名，是不可定位的。）请注意，与 CDS 名称相似，DNS 名称不具有类型；即它们不由属性类型和值对组成。

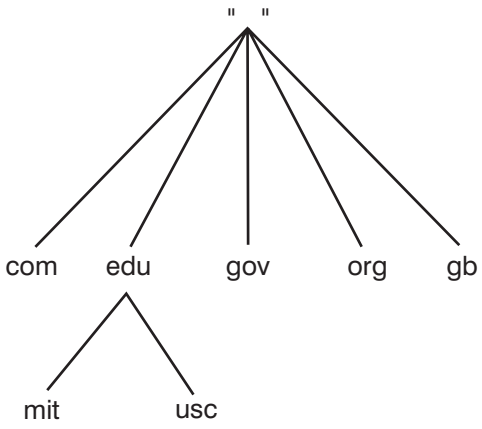


图 10. BIND 名称空间的样本部分

DNS 名称包含了由 **.**（点）分隔的分层名称字符串，并从右到左，从名称空间的根开始排列。例如，名称 **ai.mit.edu** 表示由麻省理工学院人工智能系拥有的名称空间分支。请注意，名称中元素的顺序与 CDS 名称的顺序正好相反。

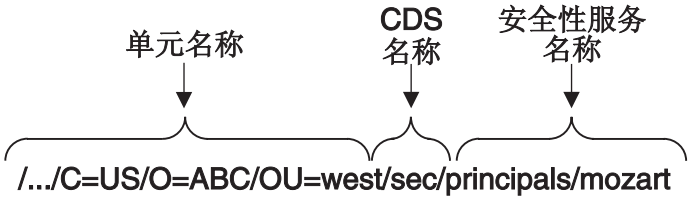
要使用 DNS 单元名作为全局 DCE 名称的一部分，原封不动地指定两个斜杠之间的 DNS 名称。例如，一个 DNS 名为 **ai.mit.edu** 的单元可能包含 CDS 名为 **/.:/profiles** 的目录。用户应该输入 **/.../ai.mit.edu/profiles** 来通过它的全局名引用目录。

“DCE 目录服务”外部的名称

不是所有的 DCE 名称都直接存储在“DCE 目录服务”中。某些服务通过一些称为*联结点*的专门 CDS 项来连接到单元名称空间中。联结点包含的绑定信息可以让客户机与目录服务以外的服务器连接。

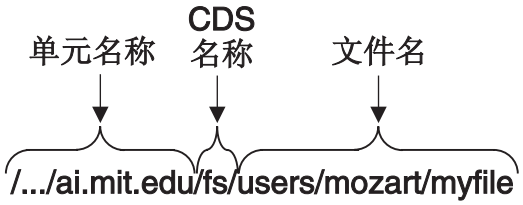
例如，安全性服务保管着有关他们的主体（用户和服务器）和信息的数据库，例如他们的口令。安全性服务联结点的缺省名称是 **/.:/sec**。

下例说明了全局 DCE 主体名的一些部分:



单元名 **/.../C=US/O=ABC/OU=west** 是 X.500 名称。**sec** 部分是 CDS 中的联结点，**principals/mozart** 是存储在安全性服务数据库中的主体名。

另一个使用联结点的服务是 DFS。DFS 文件集位置服务保管着将 DFS 文件集映射到其所在服务器的数据库。到该数据库的联结点的缺省名为 **/.:/fs**。下例说明了全局 DCE 文件名的一些部分:



全局名包含 DNS 单元名 **/.../ai.mit.edu**。**fs** 部分是 CDS 中的文件系统联结点，**/users/mozart/myfile** 是文件名。

因此，DCE 名称空间是来自许多不同源的许多种名称所连接起来的树。目录服务的 GDA 组件提供了单元外的连接以及到全局名称空间中其它单元的连接，例如 X.500 或 DNS。通过类似的方式，联结点可以让连接从单元名称空间向下传递到其它服务。

第12章 CDS 概念

单元目录服务 (CDS) 是一种高性能的分布式服务，它为单元中资源的命名和使用提供了一致的、与位置无关的方法。CDS 提供了复制 CDS 名称的能力；即，在多个节点上存储它们的副本。CDS 自动使多个副本保持一致。名称还在几个节点之间分发，这样没有一个节点需要存储所有名称。这项特性在大型单元中特别有价值。

能够复制和分发信息有许多好处，包括:

- 可用性 -- 因为可以将相同的名称存储在多个地方，所以在系统或网络故障的情况下数据往往也是可用的。
- 有效性 -- 因为可以将名称存储在最经常使用的地方的附近，所以 CDS 可以有效地查找名称。而且，一旦 CDS 找到一个名称，它就可以在所有后续查表时立即连接到这个名称。
- 负载共享 -- 因为名称位于多个地方，所以几个系统可以共享查表负载。
- 可扩展性 -- 随着网络的增长，更多应用程序使用 CDS，新名称能很容易地适应。

CDS 的工作方式

CDS 的操作包括几个主要参与者:

- 客户机应用程序
- 服务器
- 职员程序
- 中央交换库

CDS 使用客户机／服务器模型。依赖于 CDS 存储和检索其信息的应用程序就是 CDS 的客户机。客户机应用程序为它们的用户创建资源名称。通过客户机应用程序，用户可以为 CDS 提供其它信息来作为名称的属性存储。然后，当客户机应用程序用户通过其 CDS 名称引用资源时，CDS 从属性中检索数据以供客户机应用程序使用。

运行 CDS 服务器软件的系统就是 CDS 服务器。CDS 服务器存储和维护着 CDS 名称，并处理创建、修改或查找数据的请求。

称为*职员程序*的组件是客户机应用程序和 CDS 服务器之间的接口。每个 DCE 节点都必须运行一个 CDS 职员程序。职员程序接收来自客户机应用程序的请求，将请求发送给服务器，并将结果信息返回给客户机。该进程称为*查表*。职员程序还是客户机应用程序创建和修改名称所使用的接口。一个职员程序可以代表许多客户机应用程序工作。

职员程序高速缓存或保存查表的结果，使它不必为相同的信息而重复查询服务器。高速缓存被定期写入磁盘，因此信息可以在系统重新引导或应用程序重新启动时得以保留。停止 CDS 发布程序时，它停止职员程序，高速缓存被写入磁盘。高速缓存改进了性能并减少了网络流量。

第134页的图11 显示了在有 9 个节点的局域网 (LAN) 上 CDS 职员程序和服务器的样本配置。每个节点都是职员程序，CDS 服务器在两个选定的节点上运行。

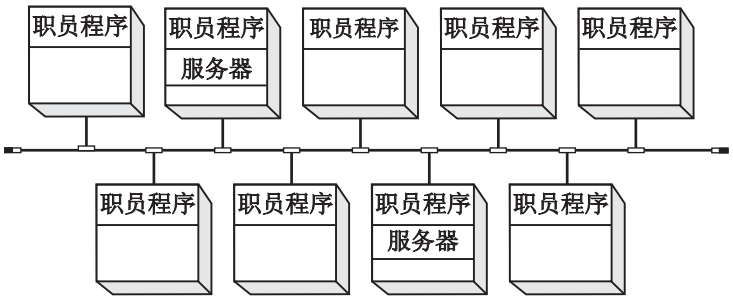


图 11. LAN 上的 CDS 职员程序和服务器

每个 CDS 服务器都有一个称为*中央交换库*的数据库，它存储着名称和其它 CDS 数据。中央交换库是 CDS 服务器代表客户机应用程序添加、修改、删除和检索数据的地方。尽管在一个服务器节点上可以存在多个中央交换库，但这不是推荐的正常配置。

图12 显示了在一个简单查表期间 CDS 客户机、职员程序、服务器和中央交换库之间的交互。 它说明了以下 CDS 查表步骤:

1. 节点 1 上的客户机应用程序向本地职员程序发送查表请求。
2. 职员程序检查它的高速缓存，如果没有在其中找到名称，则与节点 2 上的服务器联系。
3. 服务器通过检查来确定名称是否在其中央交换库中。
4. 名称存在于中央交换库中，因此服务器获得请求的信息。
5. 服务器将信息返回给节点 1 上的职员程序。
6. 职员程序将请求的数据传递给客户机应用程序。职员程序还高速缓存信息，因此在下一次客户机请求相同名称的查表时它不需要再与服务器联系。

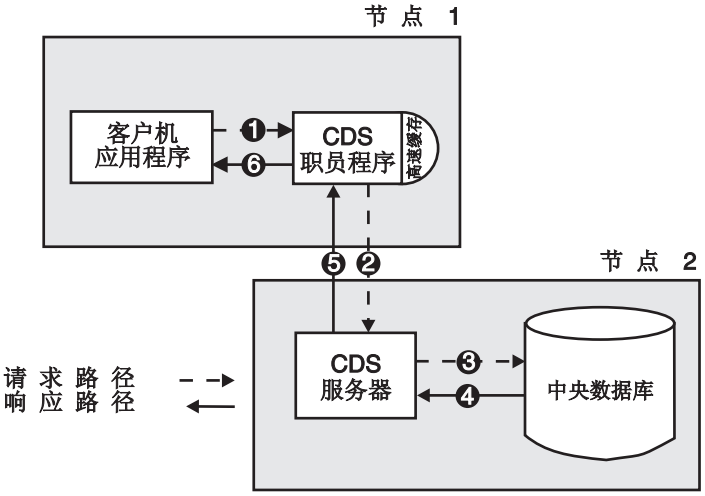


图 12. 样本 CDS 查表

副本及其内容

目录是在单元名称空间中分发和复制名称所使用单位。目录的每个物理副本，包括原始的那个，称为*副本*。创建目录的副本时，也复制了它的所有项。

副本存储在中央交换库中。可以将中央交换库看做特定服务器上目录副本的集合。在一个中央交换库中创建了目录后，可以在另一个中央交换库中创建它的副本来提高信息查表的可用性。CDS 定期地确保目录所有副本的内容保持一致。

可以有两种类型的副本存在:

- 主
- 只读

副本的类型影响对它进行的处理以及 CDS 更新它的方式。CDS 在查表或更改数据时使用的副本类型对于用户来说是不可见的。不过，它可以帮助理解两种类型之间的差异。

主副本是单元名称空间中特定目录的第一个实例。如果需要，在创建了目录的拷贝后，可以将另一个副本指定为副本。不过，每个目录中在同一时间只能有一个主副本。（请参阅第187页的『第21章 重构名称空间』 获得有关如何重新指定目录主副本的完整信息。）

主副本是目录唯一可直接修改的副本。CDS 可以创建、更改和删除主副本中的信息。因为它是可修改的，主副本比只读副本承担的开销更多，CDS 定期使用对主副本的更改来使只读副本保持最新状态。

只读副本是目录的一种拷贝，只用于查找信息。CDS 不创建、修改或删除只读副本中的名称；它只是使用对主副本的更改来更新它们。

副本可以获得三种项:

- 对象项
- 软链接
- 子指针

CDS 的首选中央交换库增强

通过将中央交换库按照客户机为 CDS 信息而联系它们的顺序进行排列，该增强提高了 CDS 客户机的性能。这可以通过使用与 CDS 客户机（相对于 CDS 服务器）位置相关的缺省值来自动实现，也可以由单元管理员手工进行重设。

该增强在以下情况下很有效：例如，由低性能 WAN 所连接的多个高性能 LAN，在每个 LAN 的中央交换库中有 CDS 副本。使用这种特性，排列最好的中央交换库是带有客户机的机器上的，后面跟着的是同一个 LAN 中带有客户机的那个。 本地中央交换库比远程中央交换库更可取。客户机只在本地中央交换库无法满足请求时使用远程中央交换库。管理员可以将缺省值重设为与中央交换库更具体的通信顺序。 通过为每个中央交换库指定排序来实现这一偏好。有关详细信息，请参阅第163页的『设置职员程序与特定中央交换库的通信』。

对象项

对象是任何一种具有 CDS 名称的真实资源--例如磁盘、应用程序或节点。创建了对象名后，客户机应用程序和 CDS 软件提供将与名称一起存储的属性。属性由属性名和值组成，描述了对象的特定操作特性。名称及其属性组成了*对象项*。当客户机应用程序请求查找名称时，CDS 返回一个或多个相关属性的值。

对象项通常是通过客户机应用程序接口来创建和管理的。例如，DCE 控制程序和 RPC 运行时的名称服务接口 (NSI) 可以让用户创建表示 RPC 服务器、组和概要的项。这些是可以让 RPC 应用程序定位和选择服务器的一些特殊种类的项。（请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Application Development Guide*了解有关 RPC 是如何使用 CDS 实现这一目的信息。）

也可以通过 DCE 控制程序 (**dcecp**) 来创建对象项。（请参阅本文档的第1页的『第1部分 DCE 控制程序』和 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference* 了解有关可以让您通过使用 **dcecp** 创建和管理对象项的命令的信息。）

每个对象都可以有一个已定义的类，它是对象项的可选属性。为它们自己的存储，使用目录服务的 DCE 组件可以为目录服务定义它们自己的对象类，并提供特定于类的属性。特定于类的属性只对与它们相关的特定对象类有意义。

中央交换库对象项代表由 CDS 预定义的一种特殊的对象类。中央交换库对象项作为指向网络中中央交换库位置的指针使用。CDS 需要该指针来查找和更新中央交换库中的数据。

创建中央交换库时，CDS 自动创建其中央交换库对象项。中央交换库对象项获得的名称和中央交换库的名称相同。中央交换库对象项由于它描述的是实际资源，因此与任何其它对象项没什么不同，但因为它只由 CDS 在内部使用，所以还是有所区别的。只能在单元根目录中创建中央交换库。因此，所有中央交换库对象项都存储在单元根目录中。CDS 自己在需要时更新和管理中央交换库对象项。它们不需要任何外部管理，除非在极少数问题解答情况下。（请在出现这些情况下寻求供应商获得帮助。）

软链接

*软链接*是一种指针，为对象项、目录或单元名称空间中的其它软链接提供了备用名称。可以通过创建从现有名称指向新名称指针的软链接来对单元的名称空间进行一些较小的重新构造。软链接还可以是提供类似多个名称的方法，这样不同类型的用户可以以对他们来说更有意义的方式来引用名称。

软链接可以是永久的，也可以在您指定的一段时间后失效。如果删除了软链接指向的名称，CDS 就自动在失效后删除软链接。

CDS 管理器应该小心使用软链接。它们不应该使用软链接来完全重新设计单元的名称空间或为希望使用对象项全名的用户提供快捷方式。对软链接的过度使用会使 CDS 名称更难于跟踪和管理。

子指针

*子指针*为单元提供了以下类型的连接:

- 单元名称空间中一个目录和其下紧接着的另一个目录之间

用户和应用程序不创建子指针；CDS 自动在有人创建新目录时创建子指针。创建子指针所在的目录是它所指向的父（上一层）目录。在尝试查找名称时，CDS 使用子指针来定位目录副本。子指针不需要管理，除非在极少数问题解决情况下。

小结

概括说来，单元由完整的一组名称组成，这些名称由单元中一个或多个 CDS 服务器共享和管理。名称可以指定目录对象项、软链接或子指针。单元名称空间的逻辑表示是目录和它们所包含的名称的层次结构。目录的每一个物理实例称为*副本*。名称物理地存储在副本中，副本又存储在中央交换库中。任何包含中央交换库并运行 CDS 服务器软件的节点就是一个 *CDS 服务器*。

图13 显示了 CDS 服务器节点的组件。每个服务器都管理着至少一个包含目录副本的中央交换库。副本可以包含对象项、软链接和子指针。该图只显示一个副本和副本中可能出现的每种类型项的一种。通常，中央交换库包含许多副本，副本又包含许多项。

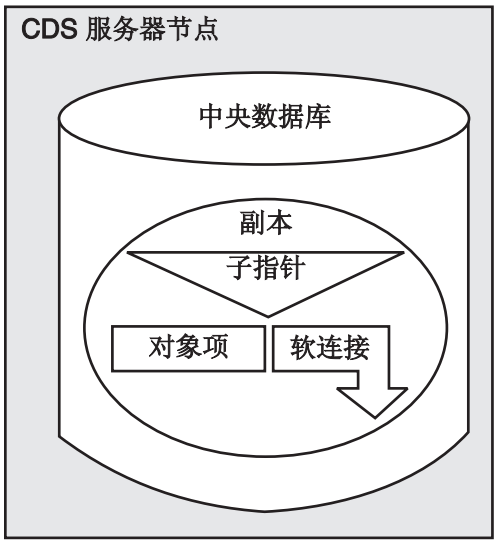


图 13. CDS 服务器节点的组件

单元目录环境中的安全性

在安全 DCE 单元操作中，服务器在通过 DCE 认证服务验证用户身份之前是不会完成用户请求的。因此，举例说，CDS 服务器只在用户身份得以验证后才可以让用户创建新目录。验证用户是否是他们所声称（的用户）的进程称为*认证*。证明是以用户名或主体名，再加上特别类型的口令的形式出现的。

CDS 服务器本身必须是经过验证的主体，这是因为以下两个原因:

- 为了向客户机证明它们是可信任的
- 为了相互证明它们有修改和管理它们所共享的数据的许可权

CDS 服务器的主体名是由配置程序自动选择的，并放在单元中包含所有 CDS 服务器的组中。该组作为 DCE 安全性服务数据库中的一项存储。在与 CDS 服务器进行了最初的联系后，职员程序通过 DCE 安全性服务确认服务器是服务器组的一个有效成员。

认证本身不是目的，而是授权进程中的一个步骤。一旦验证了主体的身份，接下来，软件就必须确定该主体是否有执行所请求操作所需的权限。这称为*授权*。因此，要创建新目录，上例中的用户不能只经过认证，还要有相应的许可权。

服务器之间需要相互认证，因为它们共享和修改复制的数据。例如，假设服务器 A 和服务器 B 都存储着同一目录的副本。将每个目录联系起来的是所有已授权维护该目录的服务器的列表。当用户修改服务器 B 处副本中的一项时，服务器 B 必须通知服务器 A 它在做这个更改。只有在服务器 B 是已认证的主体，并且是已授权修改该目录的一个主体的情况下，服务器 A 才接受更新。

CDS 许可权包括读、写、插入、删除、测试、控制和管理。根据每种许可权所关联的名称的不同，它们在含义上也略有不同，但总的来说，它们的含义如下:

- 读许可权可以让用户查看数据。
- 写许可权可以让用户添加或更改数据。
- 插入许可权可以让用户在目录中创建项。
- 删除许可权可以让用户删除项。
- 测试许可权可以让用户在不用查看任何值--即不需要对名称有读许可权--的情况下测试名称的属性是否有特定值。该许可权的主要好处是它为应用程序员检查值提供了更有效的方法: 不用读取全部值，应用程序就可以测试出是否有特定的值。
- 控制许可权可以让用户管理项的访问控制表 (ACL)。
- 管理许可权可以让用户管理目录复制。

请注意，可以为不能被认证的用户或故意要求未认证操作的用户定义特殊 ACL。在这种情况下，就不验证用户的身份，未认证用户的 ACL 项确定用户是否有权执行请求的操作。（请参阅该指南的第249页的『第6部分 DCE 安全性服务』 获得有关为未认证用户创建 ACL 的详细信息。）

CDS 用户界面

CDS 有一些*实体*，可以通过在 DCE 中提供的用户界面来管理。CDS 实体是 CDS 软件任何可单独管理的一部分。CDS 目录、软链接和对象项是使用 DCE 用户界面进行管理的最常见的实体。但某些对象项一般是通过创建它们的客户机应用程序来管理的。

DCE 控制程序提供了用于管理 CDS 实体的许多命令。该指南的第151页的『第15章 管理 DCE 目录服务』 包含了有关这些命令的信息。

其它 DCE 用户界面允许对 CDS 名称的管理进行访问。例如，用户可以通过使用 ACL 编辑器（例如 DCE 安全性服务所提供的 **dcecp acl** 对象）来控制对 CDS 目录及其内容的访问。RPC 应用程序员可以使用 **dcecp** 来在单元的名称空间中创建一些服务器项、组和配置概要。

第13章 CDS 如何查找名称

本章说明了名称与其描述的物理资源之间的关系，并解释了 CDS 如何处理查找名称的请求。理解这些概念有助于规划中央交换库和目录在单元名称空间中的位置。如果遇到查找错误或故障，它还有助于查出问题原因。请注意，本章中的图反映的并不是典型 DCE 单元名称空间的实际结构。为简便起见，图显示较少的目录和目录层次。

从名称转换到资源

就像逻辑名称空间层次结构中的目录名称转换成中央交换库中的物理副本一样，CDS 名称转换成 CDS 内部使用或由客户机应用程序使用的物理资源。名称的属性使转换成为可能。本节描述 CDS 名称与其描述的物理资源之间的关系。

第140页的图14 显示了逻辑名称空间中的三个目录及其内容，以及如何在两个中央交换库中物理实现那些目录的副本。中央交换库本身有 CDS 名称：节点 1 上的 **/.:/Paris_CH** 和节点 2 上的 **/.:/NY_CH**。后缀 **_CH** 是中央交换库的推荐命名约定。**/.:/Paris_CH** 中央交换库包含根目录和 **/.:/subsys/PrintQ** 目录的副本。**/.:/NY_CH** 中央交换库包含根目录和 **/.:/subsys** 目录的副本。建议为每个目录创建至少两个副本。因此，**/.:/subsys** 和 **/.:/subsys/PrintQ** 目录都还至少需要复制到单元中的另一个中央交换库中。

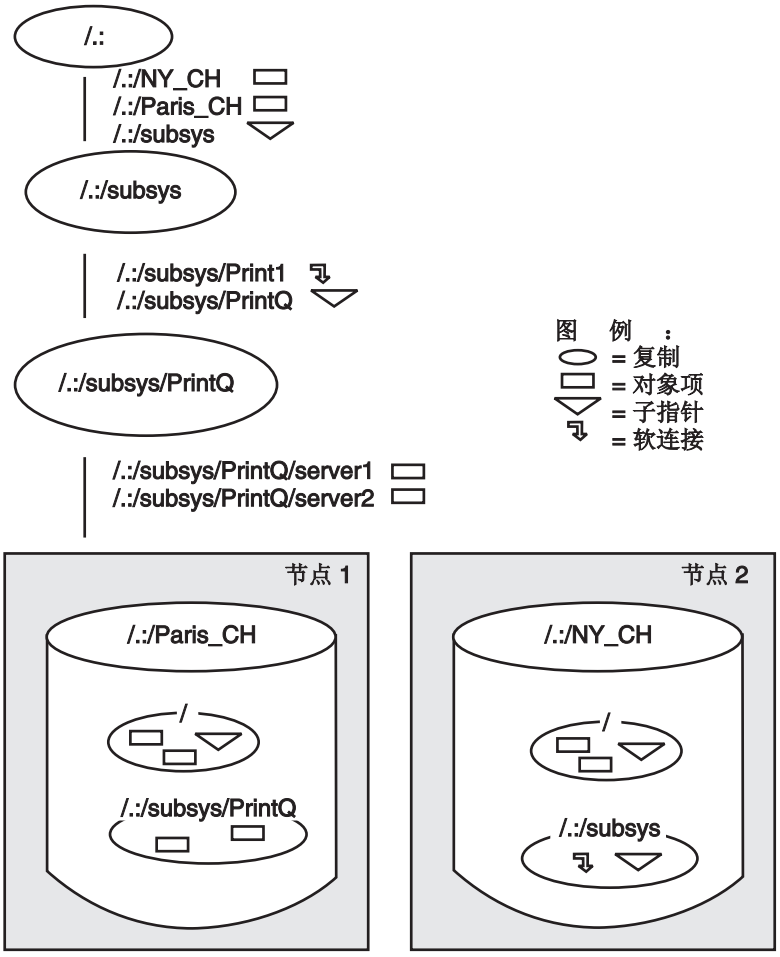


图 14. 名称空间的逻辑和物理视图

要发现资源的物理位置，CDS 将查找与其名称相关联的属性。第141页的图15 到第143页的图17 说明了各种 CDS 名称与它们所描述的资源之间的关系。这些图以图14 中的名称空间为基础。第141页的图15 到第143页的图17 中的所有名称都在同一个名称空间中，使用前缀 **/.:** 来表示单元根目录就是例证。（请参阅 第197页的『第22章 管理单元间命名』 以获取跨多个单元的名称解析的信息。）

第141页的图15 显示了两个中央交换库对象项和它们所描述的中央交换库之间的关系。中央交换库对象项与其它类型的对象项不同，因为它是由 CDS 软件创建、使用和维护的，而不是客户机应用程序。但是，它像其它任何对象项一样，它描述了网络中的物理资源：中央交换库。当您创建并命名中央交换库时，CDS 会自动创建对象项。

第141页的图15 显示了两个中央交换库对象项：**/.:/Paris_CH**，指向节点 1 上名为 **/.:/Paris_CH** 的中央交换库，和 **/.:/NY_CH**，指向节点 2 上名为 **/.:/NY_CH** 的 中央交换库。每个中央交换库对象项都有一个叫作 **CDS_CHLastAddress** 的属性，它的 **Tower** 子属性包含了 RPC 绑定信息，CDS 使用该信息来联系中央交换库驻留的节点。（请参阅附录 B 以获取 CDS 属性及其描述的列表。）

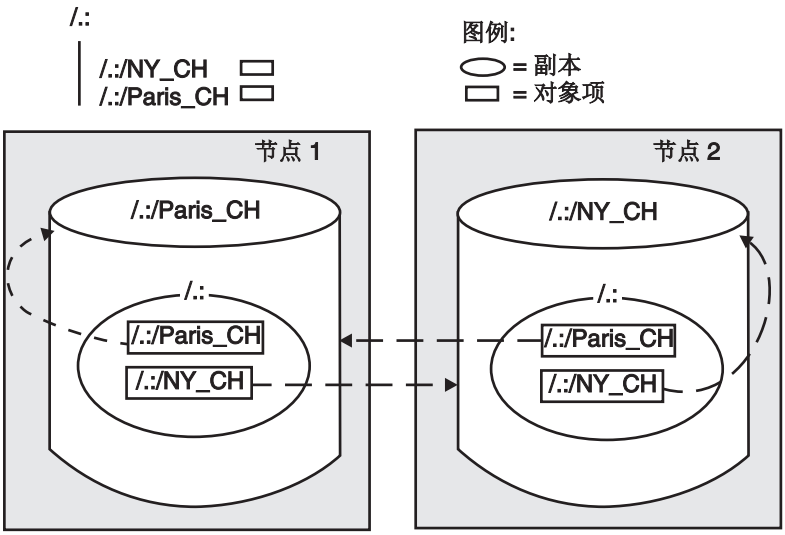


图 15. 中央交换库对象项和中央交换库

第142页的图16 显示了软链接、它指向的对象项以及对象项描述的资源之间的关系。软链接 **/.:/subsys/Print1** 有一个叫作 **CDS_LinkTarget** 的属性，该属性包含了链接指向的名称: 名为 **/.:/subsys/PrintQ/server1** 的对象项。该对象项描述了 **PrintQ** 应用程序使用的打印服务器设备。包含 **/.:/subsys/PrintQ/server1** 对象项的副本在 **/.:/Paris_CH** 中央交换库中。对象项有一个属性叫作 **CDS_Towers**，它的 **Tower** 子属性包含了 RPC 绑定信息，该信息可以使 **PrintQ** 应用程序联系打印服务器设备。

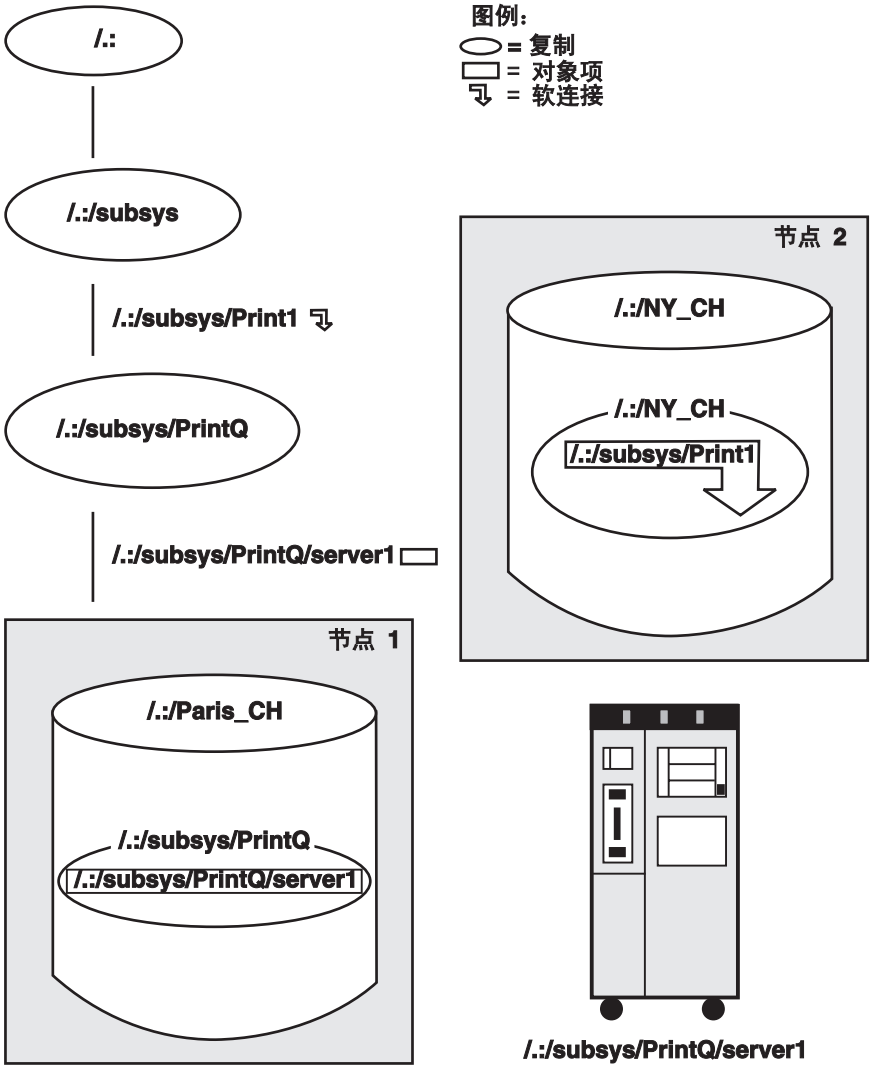


图 16. 软链接及其解析

第143页的图17 显示了目录与其关联子指针之间的关系。它说明了虽然子指针与其关联的目录同名，但子指针是名称空间中的独立项，并驻留在它指向的目录的父目录中。

这两个中央交换库的 root 副本包含了 **/.:/subsys(:)** 目录的子指针。**/.:/subsys** 子指针有一个 **CDS_Replicas** 属性，它包含了 **/.:/NY_CH** 中央交换库的名称和地址，**/.:/subsys** 目录的副本就存在于这个中央交换库中。

在 **/.:/NY_CH** 中央交换库中，**/.:/subsys** 目录的副本包含了 **/.:/subsys/PrintQ** 目录的子指针。子指针的 **CDS_Replicas** 属性包含了 **/.:/Paris_CH** 中央交换库的名称和地址，该中央交换库中有 **/.:/subsys/PrintQ** 目录的副本。

一般情况下，当一个目录有多个副本时，**CDS_Replicas** 属性会列出所有在该目录中有副本的中央交换库。可以使用 **dcecp directory show** 命令并选用 -replica 和 **-clearinghouse** 选项来显示这个属性。

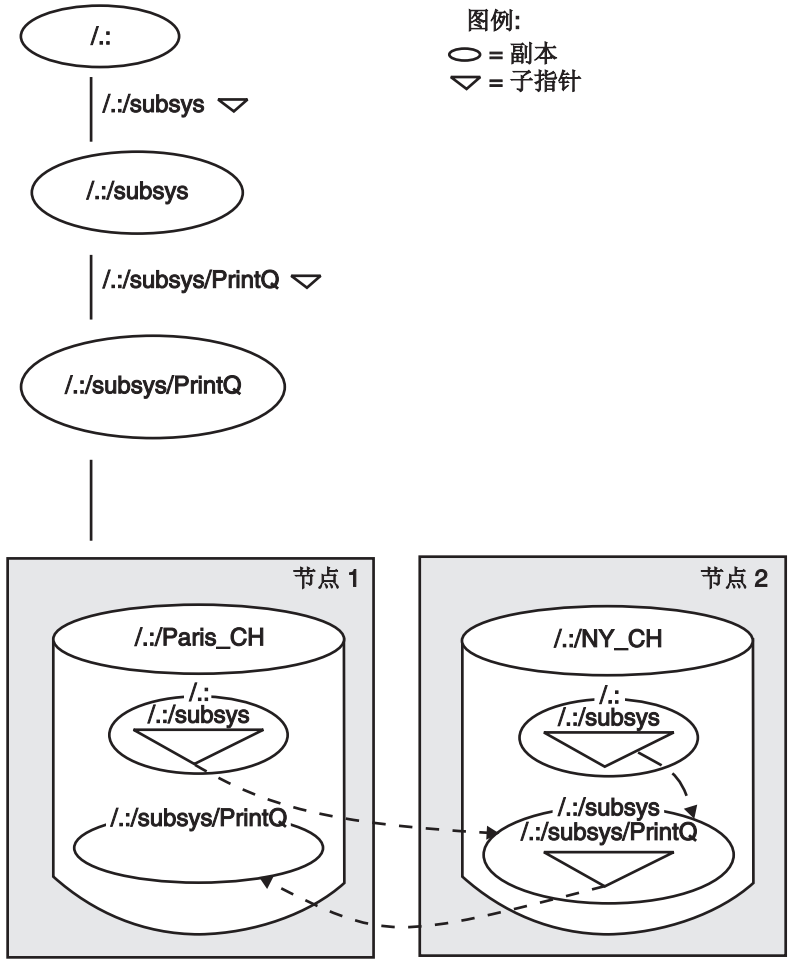


图 17. 子指针和目录

CDS 如何查找名称

如第140页的图14 到图17 所说明的，CDS 通过查找与资源名称相关的一个或多个属性来查找关于资源物理位置的信息。首先，职员程序必须知道如何找到名称。如果职员程序的高速缓存中没有名称，那么职员程序必须知道至少一个 CDS 服务器以搜索名称。

职员程序可以通过三种方法来了解 CDS 服务器及其位置:

- 通过请求和广告协议
- 通过常规查找
- **dcecp cdscache create** 命令

请求和广告协议

同一个 LAN 中的职员程序和服务器使用请求和广告协议进行通信。服务器以常规间隔广播消息，以告知相同 LAN 中的职员程序它的存在。广告消息包含关于服务器所属的单元的数据、服务器的网络地址和它管理的中央交换库。职员程序通过侦听 LAN 上的这些广告来了解服务器。职员程序在启动时还发送出请求消息来请求广告。

查找

在查找期间，如果中央交换库没有包含职员程序要搜索的名称，那么管理中央交换库的服务器将向职员程序提供尽可能多的数据，以告诉它在其它什么地方可以搜索名称。如果中央交换库包含的副本是要查找的全名的一部分，但又不是包含目标简名的副本，那么它将返回它拥有的副本中相关子指针中的数据。数据帮助职员程序在路径中查找接近目标简名的下一个子目录。子指针的 **CDS_Replicas** 属性包含该数据，其形式为中央交换库名称和绑定信息。

dcecp cdscache create 命令

DCE 管理员可以运行 **dcecp cdscache create** 命令在职员程序的高速缓存中创建关于服务器的知识。如果服务器和职员程序由广域网 (WAN) 分隔，因而使职员程序不能通过 LAN 上的广告了解服务器，此时这个命令很有用。

图18 是职员程序如何从单元名称空间的 root 向下查找对象项的示例。对象项 **/.:/Sales/Spell** 描述了位于公司的伦敦销售总部的拼写检查服务器。

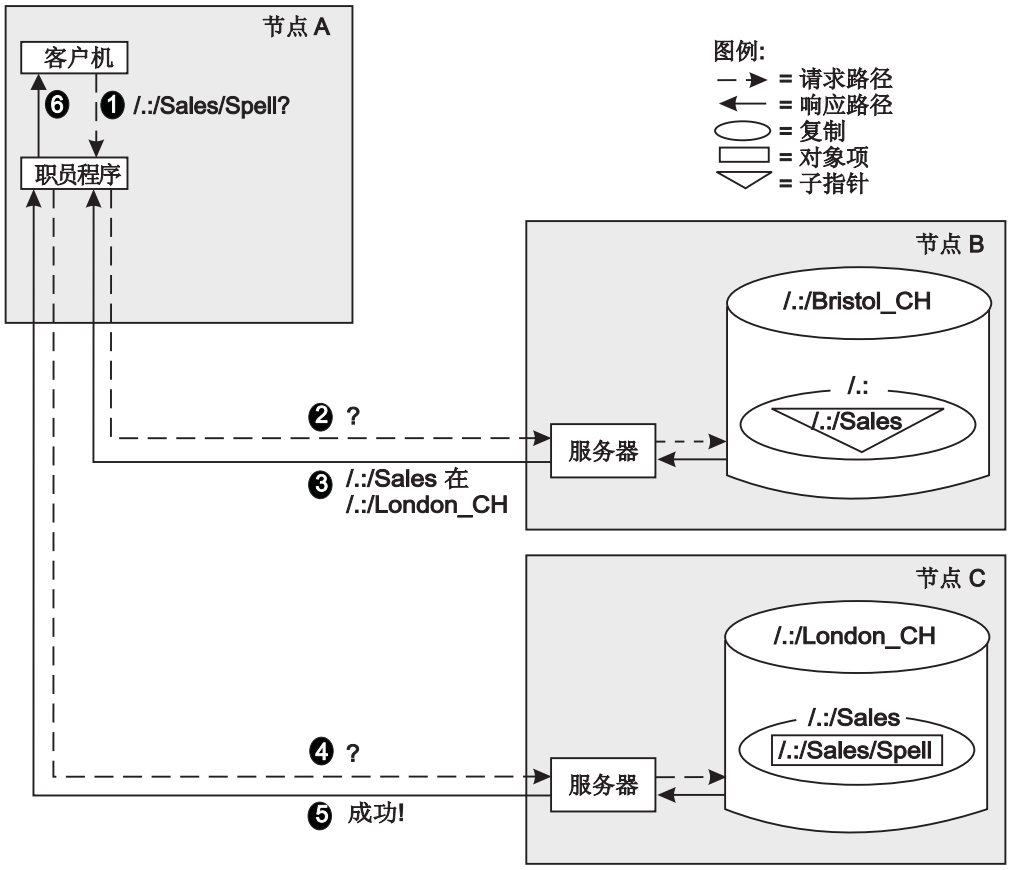


图 18. 职员程序如何查找名称

如图18 所示，职员程序通过执行以下步骤来查找期望的对象项:

1. 在节点 A 上，拼写检查应用程序请求 **/.:/Sales/Spell** 服务器的网络地址。职员程序的高速缓存中并没有这个名称，它所了解的中央交换库是节点 B 上的 **/.:/Bristol_CH** 中央交换库。
2. 职员程序使用查找请求联系节点 B 上的服务器。

3. **/.:/Bristol_CH** 中央交换库并不包含目标对象项，但它包含了根目录的副本。根据根目录中的 **/.:/Sales** 子指针，职员程序可以了解到如何联系包含 **/.:/Sales** 目录副本的中央交换库。节点 B 上的服务器将此数据返回给职员程序，通知它 **/.:/Sales** 的副本在节点 C 的 **/.:/London_CH** 中央交换库中。
4. 职员程序使用查找请求联系节点 C 上的服务器。
5. 节点 C 上中央交换库中的 **/.:/Sales** 副本包含了 **/.:/Sales/Spell** 对象项，于是服务器将拼写检查检查服务器的地址传递到职员程序。
6. 职员程序将信息返回给客户机应用程序，现在该应用程序可以远程调用拼写检查服务器。

如第144页的图18 所示，在职员程序建立了高速缓存并更了解中央交换库及其内容之后，一般情况下，查找时间通常不会很长。然而，该图显示了初始查找可能涉及到的资源和连接。该图还说明了维护名称空间中父目录和子目录之间的连通性的重要性。如果目录路径中断或不能连接中央交换库，那么职员程序就不能找到名称。

第14章 CDS 如何更新数据

只要名称在名称空间中存在，具有适当访问权的用户就能对与名称关联的数据进行更改。对 CDS 数据的任何添加、修改或删除仅在一个副本中发生：主副本。本章介绍 CDS 保持与其它副本一致性的主要方法：*更新传播*和*藏匿*操作。还描述了帮助确保 CDS 数据内一致性的两个时间戳记。通过理解本章中的概念，可以更有效地规划目录的内容和复制。

更新传播

更新传播是对在其中刚做了更改的目录的所有副本施加更改的一种直接尝试。它的主要优点是以高效及时的方式发送每次更改。

但是，与藏匿操作不同，更新传播不能确保所作的更改立即 在所有副本中生效。如果特定的副本不可用，更新传播不会失败； 更改将稍后在该副本中进行。藏匿操作确保当副本再次可用时，它与它的集合中的其它副本保持一致。

可以通过调整称为 **CDS_Convergence** 的目录属性来调整 CDS 在尝试更新传播中使用的持久性度。汇集还会影响目录上的藏匿频率。（有关查看和更改目录的汇集的详细信息，请参阅第169页的『第18章 管理 CDS 目录』 。）

藏匿操作

藏匿操作是更新集合的周期分发。它的主要功能是确保副本接收在更新传播期间无法到达它们的更改，并从名称空间除去过时的信息。

藏匿维护功能包括下列内容:

- 除去失效的软链接。当创建软链接时，可以指定失效时间。
- 维护子指针，包括除去指向已删除的目录的指针。
- 除去已删除副本的信息。

CDS 个别地藏匿每个目录。在藏匿期间，CDS 收集自最后一次成功藏匿以后所有对主副本所做的更改，并将改变分布到目录的所有只读副本中。为使藏匿成功，所有副本都必须可用。如果 CDS 无法联系一个副本，它将继续在能联系到的副本中进行更改，同时生成一个事件，通知您它无法更新的副本。然后，CDS 将定期重新尝试藏匿直到它成功完成。

藏匿可以下列三种方式之一开始:

- CDS 管理器可以输入一个命令，在目录上启动一个直接藏匿。
- CDS 启动一个藏匿作为其它名称空间管理活动的间接结果，包括下列内容:
  - 添加或除去副本
  - 创建或删除目录
  - 重新设计副本类型
  - 在父代单元中添加或删除子单元名称

  所有这些活动都在名称空间结构中产生更改；所以直接藏匿确保能尽可能快地通过名称空间反映新的结构。

- CDS 服务器在称为*后台藏匿时间*的日常间隔处自动启动藏匿。

  不考虑诸如名称管理器活动和用户造成的藏匿等其它因素，后台藏匿时间间隔保证目录藏匿之间的最大间歇时间。CDS 服务器要在其清除库中定期检查每个主副本，且在自上次成功地藏匿该目录之后又在同一目录中进行了更改的情况下，启动藏匿操作。

时间戳记如何帮助保持数据一致性

CDS 使用一些时间戳记来帮助确保数据的一致性和准确性。下列两个时间戳记存在于每个项中:

- 创建时间戳记 (CTS)
- 更新时间戳记 (UTS)

CDS 将 CTS 赋值给单元名称空间中的所有项: 清除库、目录、对象项、软链接和子指针。CTS 是反映日期、时间和位置的唯一值，其中在目录中创建了清除库、目录或项。它由两部分组成: 时间部分和其上创建名称的节点的系统标识符部分。两个部分保证在不同节点上生成的时间戳记中的唯一性。

在将新名称或更改名称传播到创建了它的目录副本期间，每个 CDS 服务器在接受新名称之前都将检查 CTS 的有效性。

UTS 反映了对清除库、目录、对象项、软链接或子指针的任何属性所作的最近一次更改。当 CDS 服务器接收到对目录中现有项的更新时，它在接受到更新之前会检查 UTS 的有效性。

当确定是藏匿目录还是在目录中作更改时，目录和副本具有 CDS 使用的几种其它时间戳记。(有关 CDS 使用的其它时间戳记属性的详细信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference*。)

当机的服务器修改

对于 DCE 3.1 AIX 版以前的发行版，尝试与当机的清除库通信将导致长时间的超时延迟。在 DCE 3.1 AIX 和 Solaris 版中，对客户机代码作了些更改以防止延迟。可以通过在 **/etc/dce/cds.conf** 文件中进行添加来覆盖或配置这些更改。

下列是用以覆盖或配置更改的增强和 **cds.conf** 修改的摘要:

- 创建了一个新的线程，定期运行并检查高速缓存清除库的可用性。如果它发现所有清除库都标记为*确定*，它将睡眠指定时间间隔。但是，如果它发现有标记为*不确定*的清除库，它将通过 **rpc_mgmt_is_server_listening( )** 调用尝试与清除库联系。如果服务器响应了，它的状态将更改为*确定*。

  连续的检查之间的延迟间隔可以通过将行添加到表单的 **/etc/dce/cds.conf** 文件来配置:

  ```
  cds.*.ch_wait_interval: value
  ```

  其中，*value* 是等待的秒数。缺省值是 120 秒。
- 在 cdsclerk 实际联系选定清除库之前，它通过将无连接句柄放于列表前来为该清除库建立句柄列表。然后，它执行 **rpc_mgmt_is_server_listening()** 调用。这些句柄的

通信超时设置为较低值，以便无需等待很长时间。如果无法建立通信，则清除库标记为*不确定*，且选择和检查了另一个适用清除库（如果有可用的话）。

这个功能也可通过将行添加到 **/etc/dce/cds.conf** 文件来配置。格式为:

```
cds.*.process_addr_check: value
```

其中，*value* 为 1（执行检查）或 0（不执行检查）。缺省值为执行检查。

- 防止客户机与标记为“关闭”的服务器联系。（这可能在其它供应商 **rpc_mgmt** 调用无法正确工作的单元中被禁用。）

  要配置这个功能，将下列行添加到 **/etc/dce/cds.conf** 文件:

  ```
  cds.*.skip_down_ch:  value
  ```

  其中，*value* 为 1（跳过*不确定*清除库）或 0（检查*不确定*清除库）。缺省值为 1。

第15章 管理 DCE 目录服务

DCE 控制程序 (**dcecp**) 提供管理 CDS 所需的大多数命令。本章描述了 DCE 控制程序允许管理的 CDS 实体，并概述了可用于管理这些实体的命令。

有关 **dcecp** 命令的详细描述，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference*。

使用 DCE 控制程序

本指南的 第3页的『第1章 DCE 控制程序介绍』介绍了 **dcecp** 及其命令语法，因此本章不再重复这些信息。实际上，本章将描述 **dcecp** 特别为管理 CDS 提供的命令。

CDS 管理的对象

DCE 控制程序对以下表示 CDS 实体的对象进行操作:

directory
该对象表示 CDS 目录。目录可以是父目录或子目录，或者是父目录或子目录的主副本或只读副本。除了子目录之外，CDS 目录还可以包含其它 CDS 资源的软链接和对象项。

link 该对象表示 CDS 目录中的软链接。软链接指向子目录（的备用名称）、对象项或其它软链接。

object 该对象表示对象项，它是单元名称空间中出现的 CDS 资源的名称。某些对象项命名了 CDS 客户机可以访问的资源（例如，磁盘、机器或应用程序）。而其它对象项命名只能由 CDS 内部使用的资源（例如，服务器和中央交换库）。

clearinghouse
该对象表示 CDS 中央交换库。中央交换库是位于 CDS 服务器设备上由服务器使用的数据库。

cdscache
该对象表示 CDS 高速缓存。CDS 高速缓存是 CDS 职员程序在本地系统上建立的关于服务器、中央交换库和其它 CDS 资源的信息集合，以便引用。

cdsalias
该对象表示 CDS 已知的 DCE 单元别名。

注: IBM DCE V3.2 中不支持创建和连接 cdsaliases；但支持 cdsalias 删除和编目。

cds 该对象表示 CDS 服务器。

cdsclient
该对象表示 CDS 客户机。

DCE 控制程序对 CDS 的操作

表8 列出了 **dcecp** 对 CDS 对象执行的操作。

表 8. DCE 控制程序对 CDS 的操作

操作	定义
add	将子目录添加到单元名称空间中的父目录中。
catalog	显示 DCE 单元的别名或中央交换库的列表。
delete	删除单元名称空间中的对象。对象类型可以是目录、对象项、软链接、中央交换库或 CDS 单元别名。
disable	从本地机器上运行的服务器中除去 clearinghosue 的知识，或者禁用 CDS 服务器或 CDS 客户机。
discard	完全除去 CDS 客户机保存的高速缓存信息。
dump	显示 CDS 高速缓存的核心内转储。
help	显示 CDS 对象类型的帮助信息，该帮助信息描述了该对象类型执行的操作或它可以接受的操作。对象类型可以是目录、对象项、软链接、中央交换库或 CDS 高速缓存。
initiate	对指定中央交换库开始执行特定操作。
list	显示目录中所有 CDS 对象包含的名称。
merge	将一个目录的内容复制到另一个目录中。
modify	修改 CDS 对象类型的属性信息。对象类型可以是目录、对象项或软链接。
operations	显示 CDS 对象类型可以执行或可以接受的操作。对象类型可以是目录、对象项、软链接或中央交换库。
ping	检查是否所有或选中的服务器都在 DCE 单元中运行。
remove	从单元名称空间中的父目录中除去子目录。
repair	对指定中央交换库开始执行诊断操作。
show	显示 CDS 对象类型的属性信息。对象类型可以是目录、对象项、软链接或中央交换库。
synchronize	告诉子目录或父目录与其副本同步（执行藏匿）。

CDS 对象属性

每个 CDS 对象都有属性，它们是与对象相关的几条或几组数据。属性可以反映或影响对象的操作行为。某些属性只能由 CDS 创建和修改；您可以根据环境需要修改其它属性。有关特定 CDS 对象的属性的完整列表，请参考适当的参考页面。另外，对于大多数对象，可以使用 **dcecp show** 操作来显示对象的所有或特定属性的名称和值。

使用 dcecp 维护 CDS

对于 CDS 维护任务，可以使用 **dcecp**。表9 中列出了完成这一任务的 **dcecp** 命令。

表 9. 控制 CDS 的 dcecp 命令

命令	定义
cdsclient disable	停止执行 CDS 职员程序。
cds disable	停止执行 CDS 服务器。
directory modify	通过指定新的主副本，重新构造目录的副本集。

表 9. 控制 CDS 的 dcecp 命令 (续)

命令	定义
directory show	显示在 DNS 或 GDS 中创建单元项所需的信息。
cdsclient show	显示 CDS 职员程序的属性。
cds show	显示 CDS 服务器的属性。

第16章 控制对 CDS 名称的访问

本章显示关于以下 CDS 授权主题的信息:

- CDS 的 DCE 授权概述
- CDS 支持的 DCE 授权组件
- CDS 支持的 DCE 许可权
- 控制对 CDS 职员程序和服务器管理操作的访问
- 控制程序命令和必需的许可权
- 编辑有关 CDS 名称的 ACL
- CDS 服务器如何获取对名称空间的访问
- 设置新名称空间中的访问控制

CDS 的 DCE 授权概述

CDS 授权允许您控制对以下 CDS 组件的用户访问:

- 存储在名称空间中的名称，包括中央交换库、目录、对象项、软链接和子指针
- 执行需特权的 CDS 职员程序和服务器命令

通过创建 ACL 来控制对名称空间中名称的访问。ACL 包含个别 ACL 项，它们指定了您授权给用户（主体）对于与 ACL 相关的名称的许可权。您创建的ACL 项在总体上确定了哪些主体可以使用名称，以及它们可以对该名称执行哪些管理操作。

合并到所有 CDS 职员程序和服务器中的 CDS ACL 管理软件执行对进入 CDS 请求的访问检查。当主体请求对 CDS 名称的操作时，存储名称的服务器上的 ACL 管理软件将检查与该名称相关的 ACL 项。然后，该软件根据 ACL 项中授予请求主体的许可权，授权或拒绝操作。同样，当主体请求 CDS 职员程序或服务器上需特权的操作时，该系统上的 ACL 管理软件将检查与表示职员程序或服务器的主体名称相关的 ACL 项。然后，该软件根据 ACL 项中授予请求主体的许可权，授权或拒绝操作。

DCE 控制程序 (**dcecp**) 提供了命令，以添加、修改、复制、删除和显示与 CDS 名称、职员程序和服务器相关的 ACL。有关这些命令的详细，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference*。本章的其余部分描述了特别试适用于 CDS的 DCE 授权。在尝试创建或修改 CDS 名称、职员程序或服务器的许可权之前，请阅读本指南中的 第249页的『第6部分 DCE 安全性服务』以获取 DCE 授权机制的完整信息。

CDS 支持的 ACL 类型

CDS 支持以下 DCE ACL 类型:

- 对象 ACL --可以使用对象 ACL 类型来授予任何 CDS 名称（即，对象项、软链接、子指针、中央交换库和目录）以及 CDS 职员程序和服务器的许可权。当与 CDS 目录关联时，以对象 ACL 类型授予的许可权只适用于目录本身，并不适用于目录的内容或任何子目录。
- 初始对象创建 ACL --初始对象创建 ACL 类型只适用于 CDS 目录名。使用该 ACL 类型将许可权特别授予目录的将来内容，包括软链接、应用程序定义的对象项、子

指针和中央交换库对象项。使用初始对象创建 ACL 类型授予的许可权只适用于目录的将来内容，而不适用于目录本身。许可权仅由创建 ACL 项之后在目录中创建的名称继承；许可权不会传播到目录中已经存在的名称。

要编辑初始对象创建 ACL，应使用 **dcecp acl modify** 命令，并选用 **-io** 选项。

- 初始容器创建 ACL --初始容器创建 ACL 类型只适用于 CDS 目录名。使用该 ACL 类型来授予对目录的许可权，该许可权会自动传播（缺省）给以后可能在目录中创建的子目录。使用初始容器创建 ACL 类型授予的许可权仅由创建 ACL 项之后创建的子目录继承；许可权不会传播到已存在的子目录。

  要编辑初始容器创建 ACL，应使用 **dcecp acl modify** 命令，并选用 **-ic** 选项。

许可权如何传播到 CDS 目录及其内容

通过为目录创建全部三种 ACL 类型（对象 ACL、初始对象创建 ACL 和初始容器创建 ACL），不仅可以对目录本身授予访问权，而且可以对目录的将来内容和所有以后可能创建的子目录授予访问权。

例如，假设您刚创建了新的目录 **/.:/sales**。如果创建了“对象 ACL”类型的 ACL 项，它授予用户 **Smith** 对 **/.:/sales** 目录的读许可权，那么 **Smith** 可以执行以下操作:

- 读取与 **/.:/sales** 目录相关的属性
- 显示存储在 **/.:/sales** 目录中的名称

如果创建了第二个“初始对象创建 ACL”类型的 ACL 项，它授予用户 **Smith** 对 **/.:/sales** 目录的读许可权，那么 **Smith** 可以执行以下操作:

- 读取与 **/.:/sales** 目录相关的属性
- 显示存储在 **/.:/sales** 目录中的名称
- 读取与所有可能以后在 **/.:/sales** 目录中创建的名称相关的属性，除非在创建通过显式修改 ACL 禁止读取

如果创建了第三个“初始对象创建 ACL”类型的 ACL 项，它也授予用户 **Smith** 对 **/.:/sales** 目录的读许可权，那么 **Smith** 可以执行以下操作:

- 读取与 **/.:/sales** 目录相关的属性。
- 显示存储在 **/.:/sales** 目录中的名称。
- 读取与所有可能以后在 **/.:/sales** 目录中创建的名称相关的属性。
- 在所有可能以后在 **/.:/sales** 目录中创建的子目录中执行上述全部三个操作。

（请参阅本指南的第249页的『第6部分 DCE 安全性服务』以获取关于缺省 ACL 的完整信息。）

主体使用的 ACL 项类型

使用 ACL 项类型来指定为其创建 ACL 项的主体的类别。表10 中描述了这些 ACL 项类型。

表 10. CDS 主体使用的 ACL 项类型

项类型	目的
user	为凭证在本地单元中认证的个别主体指定 ACL 项。

表 10. CDS 主体使用的 ACL 项类型 (续)

项类型	目的
group	为其成员的凭证在本地单元中认证的授权组指定 ACL 项。
other_obj	为本地单元中的特定认证主体指定 ACL 项，这些主体既不是由类型 **user** 的 ACL 项命名的个别用户，也不是由类型 **group** 的 ACL 项命名的组的成员。
foreign_user	为外部单元中的授权主体指定 ACL 项。
foreign_group	为其成员在外部单元中认证的授权组指定 ACL 项。
foreign_other	为外部单元中的特定认证主体指定 ACL 项，这些主体既不是由类型 **foreign_user** 的 ACL 项命名的个别用户，也不是由类型 **foreign_group** 的 ACL 项命名的组的成员。
any_other	为以上 ACL 项类型未涉及到的认证主体指定 ACL 项。
mask_obj	指定 ACL 项，它包含替换任何主体的许可权的掩码，这些主体的凭证可能是已认证的，也可能是未认证的。
unauthenticated	为不能通过认证过程的主体指定 ACL 项。
user_delegate	为充当本地单元中认证主体的中间件指定 ACL 项。
group_delegate	为充当认证主体的中间件指定 ACL 项，该主体是本地单元中授权组的成员。
other_delegate	为充当本地单元中认证主体的中间件指定 ACL 项，这些主体既不是由类型 **user_delegate** 的 ACL 项命名的个别用户，也不是由类型 **group_delegate** 的 ACL 项命名的组的成员。
foreign_user_delegate	为充当外部单元中认证主体的中间件指定 ACL 项。
foreign_group_delegate	为充当外部单元中授权组成员的中间件指定 ACL 项。
foreign_other_delegate	为充当外部单元中认证主体的中间件指定 ACL 项，这些主体既不是由类型 **foreign_user_delegate** 的 ACL 项命名的个别用户，也不是由类型 **foreign_group_delegate** 的 ACL 项命名的组的成员。
any_other_delegate	为充当本地单元或外部单元中认证主体的中间件指定 ACL 项，任何其它类型的认证主体或组的中间件的 ACL 项未命名这些主体。

CDS 支持的 DCE 许可权

CDS 支持以下 DCE 许可权：读 (**r**)、写 (**w**)、插入 (**i**)、删除 (**d**)、测试 (**t**)、控制 (**c**) 和管理 (**a**)。每种许可权的含义都略有不同，这取决于于与其相关的 CDS 名称的类型。一般情况下，许可权定义如下:

- 读许可权--允许主体查找名称，并查看与其相关的属性值。
- 写许可权--允许主体更改与名称相关的可修改属性，ACL 除外。
- 插入许可权--允许主体在目录中创建新的名称（只能与目录项一起使用）。
- 删除许可权--允许主体从名称空间中删除名称。

- 测试许可权--允许主体测试名称的属性是否有特殊值（不能真正查看任何值）；即对该名称没有读许可权。

  测试许可权为应用程序提供了一个更有效的发来验证 CDS 属性值。应用程序可以测试特殊值是否可以显示，而不是读取完整的一组值。
- 控制许可权--允许主体修改与名称相关的 ACL 项。（请注意，修改 CDS 项的 ACL 也需要读许可权；否则，**dcecp** 和 **acl_edit** 将不能绑定到该项。）控制许可权将自动授予 CDS 项的创建者。
- 管理许可权--允许主体发出控制目录复制的 CDS 命令。管理许可权只适用于目录项。

主体在尝试对名称执行管理操作之前，需要有该名称的一些许可权。否则，在主体尝试管理操作时，CDS 不能识别名称，并将返回一个错误，表示名称不存在。如果主体有一些许可权，但这些许可权并不是执行操作所需要的许可权，CDS 将返回一个错误，表示主体没有执行操作所需的足够权利。

名称的创建者被自动授予适合于所创建的名称类型的所有许可权。例如，授予创建对象项的主体对该对象项的读、写、删除、测试和控制许可权。授予创建目录的主体对该目录的读、写、插入、删除、测试、控制和管理许可权。

注: 与其它大多数文件系统执行的安全性机制不同，在对名称执行操作时，CDS 不要求主体对名称的路径名（全名）中的所有中间元素有访问权。例如，假设对象项 **object1** 存储在 **/.:/sales** 目录中。在 CDS 中，可以授予主体对 **/.:/sales/object1** 对象项的访问权，而不必授予主体对 **/.:/sales** 目录或单元根目录 (**/.:**) 的访问权。

控制对 CDS 职员程序和服务器管理操作的访问

CDS 授权允许您控制包括 CDS 职员程序和服务器上本地管理操作的 CDS 命令的使用。每个职员程序和服务器的主体名称都存储在安全性名称空间中。包含每个职员程序和服务器的绑定信息的对象项都存储在 **/.:/hosts** 子目录的 CDS 名称空间中。服务器表示为 **/.:/hosts/***hostname***/cds-server**。职员程序表示为 **/.:/hosts/***hostname***/cds-clerk**。（名称空间中不显示瘦客户机。）

每个职员程序和服务器都维护一个单独的 ACL，该 ACL 包含了指定允许执行这些操作的主体的项。不同于和名称空间中的名称相关的 ACL，与职员程序和服务器相关的 ACL 独立存在，以提供对使用这些命令的本地控制。

每当初始化新的职员程序或服务器时，都会在职员程序或服务器上创建一个 ACL。还将创建一个初始 ACL 项，授予机器主体和名称空间授权组 (**subsys/dce/cds-admin**) 对该系统上职员程序和服务器进程的读、写和控制许可权。所有其它主体，包括认证的和未认证的，都授予读许可权。创建该 ACL 项确保了在此之后，任何作为机器主体登录到该系统的用户都有权执行需特权的职员程序或服务器 CDS 命令。

注: 已经为出于此目的而使用机器主体提供了便利，并假设帐户本身（用户名和口令）已经具有一定的安全性。名称空间管理员也许更愿意修改此方案，并代表其它个别主体或授权组授予对特定职员程序和服务器的许可权。

要编辑与 CDS 职员程序或服务器相关的 ACL，使用 **dcecp acl modify** 命令并选用 **-change** 选项。例如，要更改与节点 **orion** 上 CDS 职员程序相关的 ACL 中用户 **michaels** 的许可权，输入以下命令:

```
dcecp> acl modify /.:/hosts/orion/cds-clerk -change {user michaels rw}
dcecp>
```

请记住，职员程序和服务器也是由名称空间中的项表示的。要编辑与 CDS 职员程序或服务器的名称空间项相关的 ACL，使用 **acl modify** 命令，并选用 **-entry** 和 **-change** 选项。有关如何修改 DCE 资源的 CDS 项的 ACL 的详细指令，请参阅第161页的『编辑有关 CDS 名称的 ACL』。

控制程序命令和必需的许可权

表11 列出了针对 CDS 对象操作的所有 **dcecp** 命令，以及主体要执行命令必须拥有的许可权。

表 11. DCE 控制程序命令和必需的许可权

命令	必需的许可权
cds disable	服务器的名称空间项的删除、写和创建许可权。
cds show	服务器的名称空间项的读许可权。
cdsalias catalog	要列出其别名的单元的根目录的读许可权。
cdsalias delete	单元的根目录的 auth_info 许可权。
cdscache create	对职员程序的写许可权，用于在本地 CDS 高速缓存中创建服务器项。
cdscache delete	对职员程序的写许可权，用于删除本地 CDS 高速缓存中的服务器项。
cdscache discard	CDS 高速缓存驻留的职员程序系统上的超级用户 (root) 特权。无需 CDS 许可权。
cdscache dump	CDS 高速缓存驻留的职员程序系统上的超级用户 (root) 特权。无需 CDS 许可权。
cdscache show	对职员程序的读许可权，用于从 CDS 高速缓存中检索服务器（-server 选项）或中央交换库（-中央交换库选项）信息。
cdsclient disable	职员程序的名称空间项的删除、写和创建许可权。
cdsclient show	名称空间项的读许可权。
clearinghouse catalog	无需特殊的特权。
clearinghouse create	对服务器的写许可权，用于创建中央交换库，以及对单元根目录的管理许可权。另外，服务器主体需要对单元根目录的读、写和管理许可权。
clearinghouse delete	对要删除的中央交换库的写和删除许可权，以及对将副本存储在该中央交换库中的所有目录的管理许可权。另外，服务器主体需要对相关中央交换库对象项的删除许可权，以及对将副本存储在该中央交换库中的所有目录的管理许可权。
clearinghouse disable	对中央交换库驻留的 CDS 服务器的写许可权。
clearinghouse initiate	对中央交换库服务器的写许可权和对单元根目录的管理许可权。服务器主体需要对单元根目录的读、写和管理许可权。
clearinghouse repair	对中央交换库服务器的写许可权和对单元根目录的管理许可权。服务器主体需要对单元根目录的读、写和管理许可权。
clearinghouse show	要列出其属性的中央交换库的读许可权。

表 11. DCE 控制程序命令和必需的许可权 (续)

命令	必需的许可权
clearinghouse verify	对中央交换库服务器的写许可权和对单元根目录的管理许可权。服务器主体需要对单元根目录的读、写和管理许可权。
directory add	对要放置子指针（-member 选项）的父目录的插入许可权。
directory create	对父目录的插入和读许可权，以及对存储新目录主副本的中央交换库的写许可权。另外，服务器主体需要对新目录的父目录的读和插入许可权。
directory delete	对目录的删除许可权，和对存储目录主副本的中央交换库的写许可权。服务器主体 (**hosts/***hostname***/cds-server**) 需要对父目录的管理许可权和对指向要删除目录的子指针的删除许可权。
directory list	要列出其内容的目录的读许可权。
directory merge	对源目录和目的地目录的读许可权，以及对目的地目录的插入许可权。
directory modify	对于要添加（-add 选项）、更改（-change 选项）或除去（-remove 选项）属性或属性值的目录的写许可权。
directory remove	对子指针（-member 选项）的删除许可权，或对父目录的管理许可权。
directory show	要列出其属性的目录的读许可权。对于目录副本（-replica 选项）--对副本是其成员的目录的读许可权。对于子目录（-member 选项）--对子目录的读许可权。
directory synchronize	对目录的管理、写、插入和删除许可权。另外，服务器主体需要对目录的管理、读和写许可权。
link create	要在其中创建链接的目录的插入许可权。
link delete	链接项的删除许可权，或者存储了要删除链接项的目录的管理许可权。
link modify	要修改其属性的链接的写许可权。
link show	要列出其属性的链接的读许可权。
object create	对存储对象项的父目录的插入许可权。
object delete	对象项的删除许可权，或存储对象项的父目录的管理许可权。
object modify	对于要添加（-add 选项）、更改（-change 选项）或除去（-remove 选项）属性或属性值的对象项的写许可权。
object show	要列出其属性的对象项的读许可权。

编辑有关 CDS 名称的 ACL

要编辑 ACL，而它与子目录、中央交换库、软链接或其它 CDS 对象的 CDS 名称空间中的项相关，在使用任何 **dcecp acl** 命令时，要选用 **-entry** 选项。**-entry** 选项特别适用于具有模糊路径名的情况。某些情况下，路径名可以解析成 DCE 目录服务中的叶对象*和*一些支持 ACL 的其它 DCE 环境中的对象。在这些情况中，必须使用 **-entry** 选项来编辑 CDS 中的叶对象。不必指定该选项来编辑与实际中央交换库或目录相关的 ACL。

例如，要编辑与中央交换库 **/.:/Paris1_CH** 的 CDS 项相关的“对象 ACL”中的许可权，应输入以下命令:

```
dcecp> acl modify /.:/Paris1_CH -entry -change {unauthenticated -}
dcecp>
```

要编辑与 **/.:/Paris1_CH** 中央交换库本身相关的“对象 ACL”中的许可权，应输入以下命令:

```
dcecp> acl modify  /.:/Paris1_CH  -change {unauthenticated -}
dcecp>
```

另一个示例是软链接 **/.../eng_printer**。该软链接的目标是 **/.../boston.com/print_server**。要编辑 CDS 名称空间中的软链接叶项，输入以下命令:

```
dcecp> acl modify /.../eng_printer -change -entry \
> {group subsys/dce/cds-admin rwdtc}
dcecp>
```

CDS 服务器如何获取对名称空间的访问

CDS 服务器需要有单元根目录和低级目录的许可权，以便成功执行以下 CDS 命令:

- **clearinghouse create**
- **directory create**（适用于目录和副本）
- **directory delete**（适用于目录和副本）
- **directory synchronize**

为自动执行授予所有 CDS 服务器它们所需的许可权的进程，CDS 单元配置将在固定名称 **subsys/dce/cds-servers** 下为 CDS 服务器创建授权组。单元中初始服务器的主体名称将添加到该组，作为配置进程的一部分。创建组之后，配置进程立即代表组将所有许可权（**r**、**w**、**i**、**d**、**t**、**c**、**a**）授予新名称空间的根目录。通过指定 **subsys/dce/cds-servers** 作为每个 ACL 项中的主体，来创建对象 ACL 和初始容器创建 ACL 类型的 ACL 项。这确保了组拥有所有将来目录及其内容的完整访问权。

此后，每当在单元中配置了一个新的服务器，服务器配置进程都会自动将新服务器的主体名称添加到该组。在这个进程中，单元中的所有 CDS 服务器都会接收到对名称空间中目录的适当许可权。

在新的名称空间中设置访问控制

应该规划一个一致的访问控制策略，并准备在配置第一个 CDS 服务器之后，且在创建或填充任何新的目录之*前*，尽快实现该策略。可以执行的任务如下

- 将成员添加到名称空间授权组
- 创建附加授权组
- 为未认证主体建立的最大许可权

将成员添加到名称空间授权组

为使管理名称空间和故障排除变得更容易，单元名称空间在固定名称 **subsys/dce/cds-admin** 下创建了一个名称空间授权组。然后配置进程授予该组对单元根目录的完整访问权。随着名称空间的发展，这个访问权将遍及整个名称空间。

刚创建时，授权组只包含单元配置进程期间初始名称空间管理员指定的名称。可以使用 **dcecp group add** 命令来添加公司中想要管理名称空间和排除故障的其他个人的主体名称。因为该组拥有整个名称空间的完整访问权，在必要时，其成员可以参与解决拥有较少许可权的名称空间用户遇到的问题。通过从组中除去某个用户的主体名称，该主体描述的用户将失去分配给该组的访问权。

（请参阅本指南的第249页的『第6部分 DCE 安全性服务』以获取如何添加和删除组成员的完整信息。）

创建附加授权组

授权组可以提供一种方便灵活的方法来控制对名称空间的访问权。可以根据公司、工作类型、安全性状态等组合用户，然后授予每个组对于特定目录或名称空间中其它名称的一组特定许可权。

要在本地授予权限，可以为要在名称空间中创建的每一个功能目录创建一个权限组。例如，可以创建权限组 **subsys/dce/sales-admin**，并包括负责管理 **/.:/sales** 目录的个人作为成员。每个本地权限组对它负责的目录的内容拥有完整访问权。

为未认证主体建立的最大许可权

如果要为所有未认证主体应用一组涉及整个名称空间的最大许可权，应该在配置第一个 CDS 服务器*之后*，并且在单元目录下创建和填充任何目录之*前*，就这样做。通过使用对象 ACL 和初始容器 ACL 类型来为单元 root 创建未认证 ACL 项和 **any_other** 项，随着未认证项的发展，它就可以自动传播到整个名称空间。

第17章 管理职员程序、服务器和中央交换库

CDS 职员程序、服务器和中央交换库最初都是作为 CDS 职员程序和服务器配置的一部件而创建和启动的。然后，通过手工执行一系列命令或由系统的启动脚本创建并启动职员程序和服务器进程。这些 CDS 实体都是自我调整的，除例程监控外，很少需要管理。

本章说明了如何监控 CDS 职员程序、服务器和中央交换库，以及执行其它管理任务，如备份名称空间信息。

监控职员程序、服务器和中央交换库计数器

每个职员程序、服务器和中央交换库都维护一组 *counters* 属性以跟踪自上次启动后它执行的或对它执行的读、写和其它操作。可以监控这些计数器来确定网络上生成的 CDS 流量的类型和量。

IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference 中详细描述了职员程序、服务器和中央交换库。

显示职员程序计数器

使用 **dcecp cdsclient show** 命令来显示职员程序的当前计数器值。例如，要显示与职员程序相关的所有属性的当前值，输入以下命令:

```
dcecp> cdsclient show /.:/hosts/<hostname>/cds-clerk
```

显示服务器计数器

使用 **dcecp cds show** 命令来显示服务器的当前计数器值。例如，要显示与服务器相关的所有属性的当前值，输入以下命令:

```
dcecp> cds show /.:
```

显示中央交换库计数器

使用 **dcecp clearinghouse show** 命令并选用 **-counters** 选项来显示指定中央交换库的当前计数器值。例如，以下命令显示了与远程中央交换库 **/.:/Paris1_CH** 相关的所有属性的当前值:

```
dcecp> clearinghouse show /.:/Paris1_CH  -counters
```

设置职员程序与特定中央交换库的通信

首先应该对每个中央交换库指定一个等级。等级是一个 16 位无符号整数（范围从 0 到 65535）。小的数字优先于大数字（等级 65535 *表示永远不使用这个中央交换库*）。这些等级在文本首选项文件 **$DCELOCAL/etc/cds_serv_pref** 中指定。该文件的格式是文件的每一行有一个中央交换库和一个等级。将忽略空行和注释（行尾有 #）。等级可以是 0 到 65535 (0x0000-0xFFFF)，并且可以指定为十进制、八进制（以 **0** 开头）或十六进制（以 **0x** 开头）。中央交换库名称可以是以下任何格式:

```
/.../cellname/foo_ch
/foo_ch
foo_ch
/.:/foo_ch
```

如果没有指定中央交换库的单元名称，那么就假设它是本地单元。

```
Example file:
/.:/foo_ch  50  # most preferred clearinghouse
/.:/bar_ch  100
/.../mycellname/baz_ch 100
```

如果在首选项文件中没有提到某个中央交换库，那么将会为它计算等级（因此，只需要为那些要覆盖其缺省等级的中央交换库指定等级）。缺省等级根据 IP 地址来计算:

- 地址与本地主机地址匹配的中央交换库的缺省等级是 5000。
- 与本地主机在同一个 IP 子网上的中央交换库的缺省等级是 20000。
- 与本地主机在同一个 IP 网络上的中央交换库的缺省等级是 30000。
- 其它所有中央交换库的缺省等级是 40000。

在 **cdsadv** 启动时将读取中央交换库首选项文件，并将高速缓存其中的值。如果更改等级值，必须停止 cds 客户机，除去高速缓存，然后重新启动 **cds** 客户机。

现在，以下命令将包括等级属性:

```
dcecp -c cdscache show -clearinghouse /.:/foo_ch
cdscp show cached clearinghouse /.:/foo_ch
```

监控职员程序与特定中央交换库的通信

每个 CDS 职员程序都维护一组单独的中央交换库计数器，以跟踪它定向到与其通信的每个中央交换库的读、写和其它操作。这些记录共同表示某特定职员程序的高速缓存的中央交换库实体。

可以监控职员程序的高速缓存中央交换库计数器，这样可以查看职员程序的事务在它使用的每个中央交换库中的分发情况，以及发现职员程序的请求最常定向到哪里。要这样做，使用 **dcecp cdscache show** 命令并选用 **-clearinghouse** 选项。例如，要显示 **/.:/NY1_CH** 中央交换库的本地职员程序维护的高速缓存中央交换库计数器，输入以下命令:

```
dcecp> cdscache show  /.:/NY1_CH -clearinghouse
```

显示中央交换库的内容

使用 **dcecp clearinghouse show** 命令来显示存储在特定中央交换库中的所有目录的目录名称。例如，要显示存储在中央交换库 **/.:/Chicago2_CH** 中的目录的名称，输入以下命令:

```
dcecp> clearinghouse show /.:/Chicago2_CH
```

（有关显示中央交换库信息的更多示例，请参阅第177页的『第19章 查看名称空间的结构和内容』。）

强制中央交换库检入到磁盘

在正常操作中，服务器会定期将中央交换库从内存检入到磁盘。但是，如果有服务器的写许可权，那么输入 **dcecp clearinghouse initiate** 命令并选用 **checkpoint** 选项就可以立即执行该任务。例如，要将中央交换库 **/.:/Boston3_CH** 从内存中检入到磁盘，输入以下命令:

```
dcecp> clearinghouse initiate /.:/Boston3_CH -checkpoint
```

禁用职员程序和服务器

当需要执行诊断或故障排除工作，而这些工作要求暂挂活动的职员或服务器进程，那么有时可能必须禁用在特定系统上运行的职员程序或服务器。通常，可以使用 **dce_config** 过程来启动 和停止 DCE 守护程序。通过使用 **dcecp** 命令 **cdsclient disable** 和 **cds disable** 可以禁用 CDS 职员程序和服务器。

禁用职员程序

要禁用本地节点上的职员程序，输入以下命令:

```
dcecp> cdsclient disable /.:/hosts/<hostname>/cds-clerk
```

禁用服务器

要禁用本地节点上的服务器，输入以下命令:

```
dcecp> cds disable /.:/hosts/<hostname>/cds-server
```

重新启动职员程序和服务器

CDS 职员程序和服务器进程由重新引导主机系统时执行的启动脚本自动创建和启动。但是，有时如果职员程序或服务器在重新引导时自动启动失败，或者要重新启动为在主机系统上执行备份或诊断工作而禁用的职员程序或服务器，可能需要自己运行这些脚本。**start.dce** 开始用于启动 DCE 守护程序。

重新启动职员程序

要重新启动职员程序，执行以下这些步骤:

1. 以超级用户 (**root**) 登录到职员程序系统。
2. 输入以下命令查看 **dced** 进程是否正在运行:

   ```
   # ps -e
   ```

3. 如果 **dced** 进程出现在活动进程的列表中，继续第 4 步。如果 **dced** 进程没有出现在活动进程的列表中，输入以下命令来启动进程:

   ```
   # dced
   ```

4. 输入以下命令来启动 **cdsadv** 进程:

   ```
   # cdsadv
   ```

重新启动瘦客户机配置中的职员程序

要重新启动瘦客户机配置中的职员程序，执行以下这些步骤:

1. 以超级用户 (**root**) 登录到职员程序系统。
2. 输入以下命令来启动 **cdsclerk** 进程:

   ```
   # cdsclerk -n
   ```

重新启动服务器

要重新启动服务器，执行以下这些步骤:

1. 以超级用户 (**root**) 登录服务器系统。
2. 输入以下命令来查看 **dced** 进程是否正在运行:

   ```
   # ps -e
   ```

3. 如果 **dced** 进程出现在活动进程的列表中，继续第 4 步。如果 **dced** 进程没有出现在活动进程的列表中，输入以下命令来启动进程:

   ```
   # dced
   ```

4. 输入以下命令来查看 **cdsadv** 进程是否正在运行:

   ```
   # ps -e
   ```

5. 如果 **cdsadv** 进程出现在活动进程的列表中，继续第 6 步。如果 **cdsadv** 进程没有出现在活动进程的列表中，输入以下命令来启动进程:

   ```
   # cdsadv
   ```

6. 输入以下命令来重新启动服务器:

   ```
   # cdsd
   ```

服务器进程启动时，系统上的所有中央交换库都可用。

在服务器系统升级期间保留中央交换库

如果打算升级 CDS 服务器系统上的操作系统软件，并且要保留系统上的中央交换库（或多个中央交换库），执行以下过程:

1. 确保禁用职员程序和服务器。
2. 执行系统升级之前，备份以下 CDS 文件:
   - **$DCELOCAL/etc/cds_attributes**
   - **$DCELOCAL/var/adm/directory/cds/cds_files**
   - **$DCELOCAL/var/adm/directory/cds/*_ch.checkpoint** *nnnnnnnn*
   - **$DCELOCAL/var/adm/directory/cds/*_ch.tlog** *nnnnnnnn*
   - **$DCELOCAL/var/adm/directory/cds/*_ch.version**
   - **$DCELOCAL/var/directory/cds/cds_cache.** *nnnnnnnn*
   - **$DCELOCAL/var/directory/cds/cds_cache.version**
   - **$DCELOCAL/var/directory/cds/cds_cache.wan**

   （有关所有 CDS 文件的完整路径名，请参阅 *IBM DCE 版本 3.2 AIX 和 Solaris 版: 管理指南--简介*和 *OSF DCE Porting and Testing Guide*。）
3. 执行系统升级。
4. 恢复第 2 步中备份的所有文件。
5. 要重新启动服务器，执行第165页的『重新启动职员程序和服务器』中描述的过程。服务器进程启动时，它自动查找适当的已恢复文件，并启动系统上的所有中央交换库。

备份名称空间信息

因为更新和藏匿目录可以不同时发生，又因为名称空间的分布式性质，不是总依靠传统备份方法来保存 CDS 数据。

本章的其余部分讲述了何时使用以下备份机制:

- 目录复制
- 操作系统备份

使用复制来备份名称空间信息

目录复制始终是备份名称空间中信息的最可靠方法。当在中央交换库中创建目录的新副本时，不仅在分发信息，而且在创建信息的最新、实时备份。如果一个中央交换库中的副本变为不可用，用户可以在其它中央交换库中的目录另一个副本中查找他们所需的信息。创建的目录副本越多，那么可以在名称空间中查找目录中信息的用户也就越多。

如果整个中央交换库都遭到毁坏，可以通过创建新的中央交换库，并为存储在其中的目录创建新的副本来恢复它。（有关如何创建副本的完整信息，请参阅第169页的『第18章 管理 CDS 目录』。）

使用操作系统备份

因为名称空间是分布式数据库，它的修改是不定期同步的，因此特定系统的传统备份总是包含旧的、不完整的信息。如果经常创建、修改或删除名称，恢复已作废的备份可能会使新创建的名称消失，撤销新的修改或者最近删除的名称重新出现在名称空间中。传统备份反映中央交换库情况的程度完全取决于以下条件:

- 备份已创建多久
- 自那时起做了什么修改
- 包括中央交换库文件的备份是否在目录 *dcelocal*/**var/directory/cds** 中

如果决定使用操作系统备份，只需要备份其中央交换库存储了目录主副本的服务器系统。要确保完整备份名称空间，请检查以下事项:

- 已使用 **dcecp cds disable** 命令禁用这些系统上的服务器。
- 根目录 *dcelocal*/**var/directory/cds** 中的文件已包括在备份中。

如果名称空间很小，在一个中央交换库中就可以维护，可以放心地使用传统操作系统备份来保存和恢复中央交换库数据。如果只有一个中央交换库，那么每个目录就只有一个副本（主副本）。这就无需考虑多个目录副本之间存在的差异。请记住，备份中央交换库数据越频繁，需要恢复的信息就越新。

改进重新主控的性能

当用户将目录主本重定向到新的中央交换库，如果源中央交换库（当前主本）上的时间戳记比目标中央交换库（新主本）上的时间戳记晚，那么可能需要很长时间来完成该操作。要改进重新创建主副本的性能，确保源中央交换库上的时间戳记比目标中央交换库上的时间戳记早。

第18章 管理 CDS 目录

如果管理 25 节点以下的小型缓慢增长的网络中的一个名称空间，则可以在根目录下维护所有名称，也许不必创建附加目录。但是，如果管理 25 节点以上的网络中的名称空间，则应该考虑在根目录下创建至少一个附加级别的目录。

本章说明了如何在单元名称空间中创建目录层次结构，并描述了与管理目录相关的任务，如:

- 创建和删除目录副本
- 藏匿目录
- 修改目录的收敛性

创建目录

通过创建目录，可以根据使用地点、使用频率和使用者来复制和管理对象项组。将对象项分组到单独目录还易于控制访问，因为它允许您利用缺省 ACL 项传播的优点。

CDS 单元配置在根目录下创建一个初始目录层次结构，这样 DCE 组件可以固定名称空间中的位置，于是它们可以在名称空间中创建对象项并编制目录。单元配置创建的目录之中有一个是 **subsys** 目录，独立软件供应商 (ISV) 可以在这个目录下创建它们自己的目录，以便存储它们的分布式应用程序使用的对象项。

或者，ISV 和名称空间的其它用户也许更愿意在根目录下根据它们自己的设计创建目录层次结构，用来存储它们的信息。

（有关单元配置建立的初始层次结构的详细信息，请参阅 *IBM DCE 版本 3.2 AIX 和 Solaris 版: 管理指南--简介*。）

创建目录的许可权

要创建目录，需要以下许可权:

- 将新目录插入父目录的许可权。
- 对存储新目录主副本的中央交换库的写许可权。
- 在其中输入 DCE 控制程序的 (**dcecp**) **directory create** 服务器系统的服务器主体必须有对新目录的父目录的读和插入许可权。

  如果该服务器包括在服务器授权组 **subsys/dce/cds-servers** 中，这些许可权应该已经就绪。如有怀疑，针对父目录使用 **dcecp acl show** 命令来验证服务器主体是否具有相应的许可权。（有关 **acl show** 命令的变量的详细信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference*。）

输入 directory create 命令

使用 **directory create** 命令来用指定的名称创建新目录（主副本）。缺省情况下，使用该命令时，CDS 将新目录的主副本与该目录的父目录的主副本存储在同一个中央交换库中。

例如，要创建目录 **/.:/sales**，并将新目录的主副本存储到根目录的初始中央交换库中，输入以下命令:

```
dcecp> directory create /.:/sales
dcecp>
```

注: 要使目录创建成功，输入命令时，新目录的父目录的主副本必须可用。

可以使用 **directory create** 命令的 **-clearinghouse** 选项将新目录的主副本存储在与父目录的中央交换库不同的中央交换库。例如，要将上一个示例中创建的新目录放到另一个中央交换库 (**/.:/Chicago1_CH**) 中，应输入以下命令:

```
dcecp> directory create /.:/sales -clearinghouse /.:/Chicago1_CH
dcecp>
```

（有关 **directory create** 命令的变量和选项的完整信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference*。）

检查新目录的 ACL 项

创建目录之后，要验证创建目录的用户和应用程序是否有适当的许可权。要这样做，对目录使用 **acl show** 命令以查看相关的 ACL 项。例如:

```
dcecp>  acl show   /.:/sales
{unauthenticated r--t-}
{group subsys/dce/cds-admin rwdtc}
{group subsys/dce/cds-server rwdtc}
{any_other r--t-}
dcecp>
```

（有关 **acl show** 命令的完整信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference*。）

如果没有从新目录的父目录处继承必要的许可权，使用 **acl modify** 命令来创建必需的 ACL 项。例如:

```
dcecp> acl modify /.:/sales -add
{user cell_admin rwdtcia}
dcecp>
```

（有关 **acl modify** 命令的变量和选项的完整信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference*。）

升级单元根目录的版本

升级单元根目录的目录版本有特殊意义。这个过程意味着单元中的所有 CDS 服务器都已经升级到最新版本，单元根目录已经复制到单元中的所有 CDS 服务器中。设置了单元根目录的 **CDS_UpgradeTo** 属性之后，服务器软件马上就认出它，并设置单元中所有目录的 **CDS_UpgradeTo** 属性。最后，单元中所有受影响的目录的 **CDS_DirectoryVersion** 属性都将升级成新的值。

升级目录的版本

要使用给定 CDS 发行版中的新功能，可能需要明确更新目录的版本。通常在复制目录的服务器全都已经升级到软件的最新版本时会发生这种情况，因为老版本不能识别新的功能。

要升级目录版本，需要有该目录的写许可权，并且必须使用以下命令:

```
dcecp> directory modify directory-name -add {CDS_UpgradeTo <v.n>} } -single
dcecp> directory synchronize directory-name
```

最终，所有包含该目录的副本的中央交换库检测是否存在 **CDS_UpgradeTo** 属性，并升级适当副本的 **CDS_ReplicaVersion** 属性。还可以对所有正在复制目录的中央交换库使用以下命令:

```
dcecp> clearinghouse verify clearinghouse-name
```

该命令强制运行服务器后台线程，从而让您可以在作业完成之前执行其它任务。验证了所有受影响的中央交换库之后，需要再次藏匿目录，将 **CSA_DirectoryVersion** 属性最终设置成适当的值。在所有副本的所有 **CDS_ReplicaVersion** 属性值都包含新的值之前，**CDS_DirectoryVersion** 属性不会升级。

创建只读副本

有时，可能要创建目录的只读副本。可能出于以下目的，要创建目录的只读副本:

- 要将目录中包含的信息分发到整个网络，并使其它位置处的用户和应用程序可以访问该信息。
- 要改进响应时间，特别是在用户分隔较远的名称空间中。应该在最常使用目录中包含的信息的用户组和应用程序附近的中央交换库中创建只读副本。
- 要保留目录主副本中信息的备份。维护多个副本确保了个别副本的临时丢失不会引起服务中断，而且可以容易地恢复丢失的副本。甚至应该将目录（存储仅在某特定站点上使用的信息）复制到服务器的另一个中央交换库中，最好是另一地方的一个服务器，这样某个站点的本地故障就不会导致两个副本同时不可用。（有关使用目录复制作为备份 CDS 信息的方法的详细信息，请参阅 第163页的『第17章 管理职员程序、服务器和中央交换库』。）

目录的只读副本不会遇到受用户改动的危险。用户可以查找只读副本中的信息，但他们不能创建新的信息或修改现有信息。

使用 **directory create** 命令并选用 **-replica** 选项来创建只读副本。如果中央交换库的用户需要访问目录，但不需要或不允许更新目录内容，那么应该在该中央交换库中创建副本。

创建副本之前

尝试创建副本之前，验证中央交换库（包含要复制的目录的主副本）是否正在运行，以及是否可以到达。要验证是否满足该条件，需执行以下步骤:

1. 对于要复制的目录，使用 **directory show** 命令来显示目录属性值，并查看 **CDS_Replicas** 属性。该属性的值显示了当前存储目录副本的中央交换库的名称。例如:

```
dcecp> directory show /.:/sales
{RPC_ClassVersion {01 00}}
{CDS_CTS 1994-08-12-09:52:30.396-04:00I0.000/00-00-c0-f7-de-56}
{CDS_UTS 1994-08-12-09:52:31.506-04:00I0.000/00-00-c0-f7-de-56}
{CDS_ObjectUUID a37d84d0-b5dc-11cd-8ffe-0000c0f7de56}
{CDS_Replicas
 {{CH_UUID ce7ed810-b5db-11cd-8ffe-0000c0f7de56}
  {CH_Name /.../Chicago1/Chicago1_CH}
  {Replica_Type Master}
  {Tower {ncacn_ip_tcp 130.105.5.16}}
  {Tower {ncadg_ip_udp 130.105.5.16}}}}
{CDS_AllUpTo 1994-08-12-09:52:31.566-04:00I0.000/00-00-c0-f7-de-56}
{CDS_Convergence medium}
{CDS_ParentPointer
 {{Parent_UUID d034bc25-b5db-11cd-8ffe-0000c0f7de56}
  {Timeout
   {expiration 1994-08-12-09:52:30.396}
   {extension +1-00:00:00.000I0.000}}
  {myname /.../Chicago1/sales}}}
{CDS_DirectoryVersion 3.0}
{CDS_ReplicaState on}
{CDS_ReplicaType Master}
{CDS_LastSkulk 1994-08-12-09:52:31.566-04:00I0.000/00-00-c0-f7-de-56}
{CDS_LastUpdate 1994-08-12-09:52:31.506-04:00I0.000/00-00-c0-f7-de-56}
{CDS_RingPointer ce7ed810-b5db-11cd-8ffe-0000c0f7de56}
{CDS_Epoch a3df2a50-b5dc-11cd-8ffe-0000c0f7de56}
{CDS_ReplicaVersion 3.0}
dcecp>
```

2. 有了此信息，使用 **directory show** 命令并选用 **-clearinghouse** 和 **-replica** 选项来验证是否可以从存储主副本的中央交换库中获取响应。例如:

```
dcecp> directory
show /.:/sales -replica -clearinghouse  /.:/Chicago1_CH
{RPC_ClassVersion {01 00}}
{CDS_CTS 1994-08-12-09:52:30.396-04:00I0.000/00-00-c0-f7-de-56}
{CDS_UTS 1994-08-12-09:52:31.506-04:00I0.000/00-00-c0-f7-de-56}
{CDS_ObjectUUID a37d84d0-b5dc-11cd-8ffe-0000c0f7de56}
{CDS_Replicas
 {{CH_UUID ce7ed810-b5db-11cd-8ffe-0000c0f7de56}
  {CH_Name /.../Chicago1/Chicago1_CH}
  {Replica_Type Master}
  {Tower {ncacn_ip_tcp 130.105.5.16}}
  {Tower {ncadg_ip_udp 130.105.5.16}}}}
{CDS_AllUpTo 1994-08-12-09:52:31.566-04:00I0.000/00-00-c0-f7-de-56}
{CDS_Convergence medium}
{CDS_ParentPointer
 {{Parent_UUID d034bc25-b5db-11cd-8ffe-0000c0f7de56}
  {Timeout
   {expiration 1994-08-12-09:52:30.396}
   {extension +1-00:00:00.000I0.000}}
  {myname /.../Chicago1/sales}}}
{CDS_DirectoryVersion 3.0}
{CDS_ReplicaState on}
{CDS_ReplicaType Master}
{CDS_LastSkulk 1994-08-12-09:52:31.566-04:00I0.000/00-00-c0-f7-de-56}
{CDS_LastUpdate 1994-08-12-09:52:31.506-04:00I0.000/00-00-c0-f7-de-56}
{CDS_RingPointer ce7ed810-b5db-11cd-8ffe-0000c0f7de56}
{CDS_Epoch a3df2a50-b5dc-11cd-8ffe-0000c0f7de56}
{CDS_ReplicaVersion 3.0}
dcecp>
```

directory show 目录以及 **-clearinghouse** 和 **-replica** 选项显示了目录及其副本角色的所有属性值。

注: 如果目录的现有副本集的任何只读副本不可用，那么就不能完成。正常的藏匿进程在目录副本集的所有副本都变成可用后立即完成复制。

创建副本的许可权

要创建副本，需要以下许可权:

- 要复制的目录的管理许可权
- 存储新副本的中央交换库的写许可权
- 如果要使副本创建成功，在其中输入 **directory create** 命令并选用 **-clearinghouse** 和 **-replica** 选项的服务器系统的服务器主体必须有要复制的目录的读、写和管理许可权。

  如果该服务器包括在服务器授权组 **subsys/dce/cds-servers** 中，这些许可权应该已经就绪。如有怀疑，使用 **acl check** 命令来验证服务器主体是否具有相应的许可权。（有关使用 **acl check** 命令的完整信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference*。）

输入 directory create 命令

使用 **directory create** 命令并选用 **-replica** 和 **-clearinghouse** 选项来创建目录的副本，并将它存储到指定的中央交换库中。例如，以下命令创建 **/.:/mfg** 目录的副本，并将副本存储到中央交换库 **/.:/Paris1_CH** 中:

```
dcecp> directory create /.:/mfg -replica -clearinghouse /.:/Paris1_CH
dcecp>
```

删除只读副本

有时，当存储副本的中央交换库的用户不再需要某个只读副本包含的信息时，可能需要删除该只读副本。准备删除副本是其成员的目录，或者在永久除去存储副本的中央交换库之前，可能也需要删除只读副本。

删除副本的许可权

要删除副本，需要以下许可权:

- 要删除其副本的目录的管理许可权。
- 对从中删除副本的中央交换库的写许可权

输入 directory delete 命令

使用 **directory delete** 命令并选用 **-replica** 和 **-clearinghouse** 选项从指定中央交换库中删除副本。例如，以下命令从 **/.:/Chicago2_CH** 中央交换库 中删除 **/.:/eng** 目录的副本:

```
dcecp> directory delete /.:/eng -replica -clearinghouse /.:/Chicago2_CH
dcecp>
```

注: 删除目录本身（使用 **directory delete** 命令）只能删除目录的主副本。（有关如何删除主副本的完整信息，请参阅第187页的『第21章 重构名称空间』。）

藏匿目录

藏匿操作是对名称空间的最新修改的定期分发。CDS 根据对目录的 **CDS_Convergence** 属性指定的值，按常规间隔藏匿每个目录。要确保更新尽快分发到目录的所有副本，可以使用 **directory synchronize** 命令来启动目录的藏匿操作，而不是等到下一次预定的藏匿再分发新的信息。可以使用该命令来执行以下任务:

- 如果不想等到下一次藏匿就分发对目录的内容或副本集的关键更新
- 藏匿服务器上存储副本的目录，这些服务器在某个扩展时期内不可操作，然后会重新联机

藏匿目录所需的许可权

要藏匿目录，必须拥有下列许可权:

- 对目录的管理、写、插入或删除许可权。
- 对于输入 **directory synchronize** 命令的服务器系统，其服务器主体要有对藏匿的目录的读、写和管理许可权。

  如果该服务器包括在服务器授权组 **subsys/dce/cds-servers** 中，这些许可权应该已经就绪。如有怀疑，使用 **acl show** 命令来验证服务器主体是否具有相应的许可权。（有关 **acl show** 命令变量的完整信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference*。）

输入 directory synchronize 命令

使用 **directory synchronize** 命令来立即藏匿目录。

输入命令后，在藏匿过程中，**dcecp** 会临时暂挂 **dcecp>** 提示符。藏匿带有较大副本集的目录可能需要一些时间。如果提示符出现，并且没有错误消息，那么藏匿成功。如果提示出现后显示了错误消息，那么藏匿失败。

要使藏匿成功，目录副本集的每个副本都必须是可用的。藏匿有时会失败，特别是如果目录有很大的副本集，或者存储目录副本的服务器在很远的位置，而网络连通性始终不可靠。

藏匿失败不会使 CDS 变成不可用。虽然藏匿进程无法更新它不能联系的副本中的信息，但它总是更新它可以访问的副本中的信息。某些情况下，一些副本包含最新信息，而另一些却不包含最新信息。藏匿失败时，CDS 会根据基于目录收敛性值的间隔自动重复藏匿进程，直到用最新更改更新了集合中的所有副本。当所有副本都包含了同样的信息时，CDS 就认为藏匿成功。

如果藏匿目录继续失败，可以通过复查存储目录主副本的服务器上的 CDS 事件日志来确定原因。例如，以下命令将藏匿 **/.:/admin** 目录:

```
dcecp> directory synchronize /.:/admin
dcecp>
```

同步 CDS 服务器时钟

对目录执行了藏匿操作之后，可能会接收到以下消息:

未同步服务器时钟

表示没有同步服务器时钟。如果是这样，应首先检查服务器系统上的系统时钟是否确实已同步。如果已同步，但仍接收到这条消息，那么个别服务器上的系统时钟也许被

错误地设置成将来时间，然后又恢复了。这将导致 CDS 的问题，因为存储在中央交换库中的时间戳记可能是无效的（任何比当前时间晚 5 分钟的时间戳记）。

如果是这种情况，应该将系统时钟调整到当前时间，然后输入以下命令:

```
dcecp> clearinghouse repair <clearinghouse-name> -timestamps
```

该命令将禁用中央交换库，分析并修复坏的时间戳记，将中央交换库检入磁盘，并重新启用中央交换库。要使用该命令，必须对中央交换库驻留的服务器拥有写许可权。另外，应该对复制需要修复的目录（及其对象）的所有中央交换库执行该命令。

执行了 **clearinghouse repair** 之后，应该可以成功藏匿目录了。

修改目录的收敛性

指定给目录 **CDS_Convergence** 属性的值确定了存储目录主副本的服务器藏匿目录副本集的频繁程度。目录的收敛性值可以设置成 **high**、**medium** 或 **low**。

收敛性值设置成 **high** 的目录每 12 小时至少藏匿一次。如果更新了目录，存储主副本的服务器会立即尝试将新的信息传播到整个副本集。如果该更新传播失败，服务器会安排在一小时内藏匿目录。如果第一次藏匿失败，那么每隔 1 小时就会执行一次藏匿，直到藏匿成功。

收敛性值设置成 **medium** 的目录每 12 小时至少藏匿一次。如果更新了目录，存储主副本的服务器会立即尝试将新的信息传播到整个副本集。如果传播失败，服务器会等到下一次藏匿时同步副本集。

收敛性值设置成 **low** 的目录每 24 小时至少藏匿一次。如果更新了目录，存储主副本的服务器会立即尝试将新的信息传播到整个副本集。如果传播失败，服务器会等到下一次藏匿时同步副本集。

每个新创建的目录都会继承其父目录的收敛性值。创建名称空间时，会自动将根目录的收敛性值指定成 **medium**。除非更改这个值，或者在创建低级目录后更改它们的收敛性值，否则在根目录下创建的所有目录的收敛性值也将是 **medium**。对于大多数目录，不需要修改这个值。但是，有时您可能会发现将目录的收敛性设置成 **high** 或 **low** 会很有用。

修改目录的收敛性之前

修改目录的收敛性之前，要验证目录的当前收敛性值。要这样做，使用 **directory show** 命令来显示目录的属性值并查看 **CDS_Convergence** 属性值。

修改目录的收敛性的许可权

要修改目录的收敛性，必须有目录的写许可权。

输入 directory modify 命令

使用 **directory modify** 命令并选用 **-change** 选项来为目录的 **CDS_Convergence** 属性指定 **high**、**medium** 或 **low** 值。例如，以下命令将 **/.:/sales/us** 目录的收敛性值设置成 **high**:

```
dcecp> directory modify /.:/sales/us -change {CDS_Convergence high}
dcecp>
```

第19章 查看名称空间的结构和内容

需要查看单元名称空间的结构和内容时，可以使用 **dcecp**。**dcecp** 通过命令行界面显示信息。本章说明了如何使用 **dcecp** 来显示名称空间信息。

列出目录的内容

DCE 控制程序 (**dcecp**) 提供了 **directory list** 命令，它允许您显示单元名称空间中目录子代的列表。目录的子代都是目录中的子指针、中央交换库、对象项和软链接。

要使用 **directory list** 命令，必须对要显示的 CDS 名称有读许可权。

对于目录内容的完整列表，输入 **directory** 命令以及要查看其内容的（一个或多个）目录名称。例如:

```
dcecp> directory list /.:/eng
/.../eng_cell.osf.org/hosts/eng/aud-acl \
/.../eng_cell.osf.org/hosts/eng/aud-svc \
/.../eng_cell.osf.org/hosts/eng/cds-clerk \
/.../eng_cell.osf.org/hosts/eng/cds-server \
/.../eng_cell.osf.org/hosts/eng/dts-entity \
/.../eng_cell.osf.org/hosts/eng/profile \
/.../eng_cell.osf.org/hosts/eng/self \
/.../eng_cell.osf.org/hosts/eng/CDS_CTS \
/.../eng_cell.osf.org/hosts/eng/CDS_UTS \
dcecp>
```

缺省情况下，**directory list** 命令显示目录中包含的对象的全名（对象名由 ***/.../pathname*** 作为前导）。要只列出对象的 RDN，输入 **directory list** 命令并选用 **-simplename** 选项。

如果只要显示某特定种类的目录子代的名称，应在 **directory list** 命令中包括适当的选项。例如，输入以下命令来显示存储在 **/.:/eng** 目录中的所有软链接的名称:

```
dcecp> directory list  /.:/eng/ -links
/.../eng_cell.osf.org/hosts/eng/CDS_CTS \
/.../eng_cell.osf.org/hosts/eng/CDS_UTS
dcecp>
```

显示 CDS 名称的属性值

要显示与名称空间中名称相关的属性的任意或所有当前值（职员程序或服务器除外），使用 **dcecp show** 操作。

show 操作的基本语法如下:

```
object-type show object-name
```

其中 *object-type* 是要显示其信息的 CDS 对象的类型，*object-name* 是以要查询的对象的简名（即，CDS 全名）结尾的完整目录规范。

要使用 **show** 操作，必须对要显示的名称有读许可权。

下例中，**show** 操作显示了与 **/.:/Chicago2_CH** 中央交换库相关的 **CDS_CHDirectories** 属性的当前值。操作返回的显示数据显示了属性的两个值，每个值都有两部分。属性

值部分是目录的 UUID 和目录名称。**show** 操作分别显示值。对于每个值，它首先在一行内列出属性名称并以冒号结尾，然后是值部分。

```
dcecp> clearinghouse show  /.:/Chicago2_CH
{RPC_ClassVersion
 {01 00}}
{CDS_CTS 1994-01-24-07:12:51.966-05:00I0.000/00-00-c0-f7-de-56}
{CDS_UTS 1994-02-03-07:17:35.794-05:00I0.000/00-00-c0-f7-de-56}
{CDS_ObjectUUID 0094e40e-bb43-1d43-9e0a-0000c0f7de56}
{CDS_AllUpTo 1994-02-03-09:17:06.393-05:00I0.000/00-00-c0-f7-de-56}
{CDS_DirectoryVersion 3.0}
{CDS_CHName /.../Chicago2/Chicago2_CH}
{CDS_CHLastAddress
 {Tower ncacn_ip_tcp:130.105.5.16[]}}
{CDS_CHLastAddress
 {Tower ncadg_ip_udp:130.105.5.16[]}}
{CDS_CHState on}
{CDS_CHDirectories
 {dir_uuid 00595ca5-bb46-1d43-9e0a-0000c0f7de56}
 {directory /.../Chicago2}}
{CDS_CHDirectories
 {dir_uuid 00888574-bb53-1d43-9e0a-0000c0f7de56}
 {directory /.../Chicago2/subsys}}
{CDS_CHDirectories
 {dir_uuid 0069ff14-bb55-1d43-9e0a-0000c0f7de56}
 {directory /.../Chicago2/subsys/dce}}
{CDS_CHDirectories
 {dir_uuid 0023cc38-bb56-1d43-9e0a-0000c0f7de56}
 {directory /.../Chicago2/subsys/dce/sec}}
{CDS_CHDirectories
 {dir_uuid 0026d57c-bb57-1d43-9e0a-0000c0f7de56}
 {directory /.../Chicago2/hosts}}
{CDS_ReplicaVersion 3.0}
{CDS_NSCellname /.../Chicago2}
dcecp>
```

在下例中，**show** 操作显示了存储在 **/.:/sales** 目录中的对象项:

```
dcecp> object show  /.:/sales
{CDS_CTS 1994-06-23-15:56:44.734+00:00I0.000/08-00-2b-0f-59-bf}
{CDS_UTS 1994-08-08-22:23:54.226+00:00I0.000/08-00-2b-0f-59-bf}
{CDS_ClassVersion 1.0}
dcecp>
```

以下命令显示了存储在 **/.:/mfg** 目录中的所有软链接:

```
dcecp> link show /.:/mfg
{CDS_CTS 1994-06-23-15:56:44.734+00:00I0.000/08-00-2b-0f-59-bf}
{CDS_UTS 1994-08-08-22:23:54.226+00:00I0.000/08-00-2b-0f-59-bf}
{CDS_LinkTarget = /.../abc/mfg/robotics_controller1}
dcecp>
```

显示职员程序和服务器属性信息

要显示与单元名称空间中职员程序和服务器项相关的属性的值，使用 **dcecp** 命令 **cds** 和 **cdsclient**。每个命令的基本语法是:

```
cds show cds-server-name
cdsclient show cds-client-name
```

要使用这些命令，必须对要显示的 CDS 名称有读许可权。

在 **show** 操作行上，允许在职员程序和服务器的简名中使用通配符。

下例中，**show** 操作显示了与本地职员程序相关的所有属性的当前值:

```
dcecp> cdsclient show /.:/hosts/hostname/cds-clerk
```

返回的显示如下:

```
{Creation_Time 1996-08-01-15:39:06.052+00:00I-----}
{Protocol_Errors 0}
{Authentication_Failures 0}
{Read_Operations 1088}
{Cache_Hits 928}
{Cache_Bypasses 157}
{Write_Operations 68}
{Miscellaneous_Operations
94}
```

第20章 使用 CDS 子树命令来重构 CDS 目录

有时因为全体重构或其它原因，需要组合或重新安排 CDS 名称空间中的各种目录或目录的子树。

例如，假设您公司的工程组 **/.:/eng** 和研发组 **/.:/rnd** 组合，并且这两个组开始共享一组公共应用程序和其它网络资源。可以通过合并这两个目录的内容来反映名称空间层次结构中的组织变化。

同样，如果工程组成为研发组的附属，则可以通过创建空目录 **/.:/rnd/eng**，然后将 **/.:/eng** 目录的内容合并到 **/.:/rnd/eng** 中，以有效地将 **/.:/eng** 附加到 **/.:/rnd** 下，来反映这个变化。

合并和附加过程概述

要合并或附加 CDS 目录，使用 DCE 控制程序 (**dcecp**) **directory merge** 命令。这两个过程的基本步骤如下:

1. 在系统提示中，输入 **dcecp** 来调用 DCE 控制程序。
2. 将一个现有目录与另一个现有目录进行合并或附加。要这样做，使用 **directory merge** 命令将关于目录子代（对象项、软链接和子目录）的信息与另一个目录的信息组合，或者将信息附加到现有最底层目录下。
3. 使用 **directory delete** 命令将第 2 步中合并的源目录或子树（及其内容）从其层次结构中的老位置处删除。使用 **link create** 命令来用一个同名软链接替换已删除的目录信息，将信息查找重定向到新的位置。

注: 在合并目录中显示中央交换库、重复名称或不可用名称需要特殊处理。以下部分中描述的 **merge** 和 **append** 操作假设在源目录和目标目录或子树中没有重复名称，并且存储受影响目录的主副本的中央交换库在启动操作时是启用的，并且可以访问。

本节中描述的示例 **merge** 和 **append** 操作都基于示例名称空间，如下图所示。

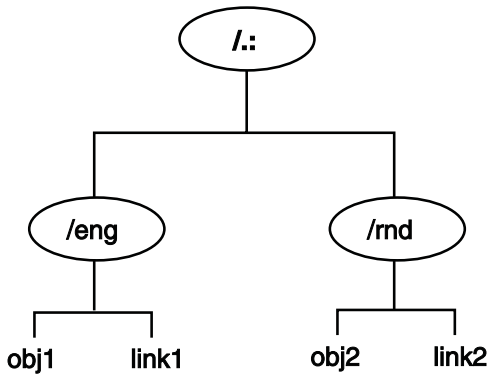


图 19. 示例名称空间层次结构

示例名称空间由根目录下的两个目录 **/.:/eng** 和 **/.:/rnd** 组成。源目录 (**/.:/eng**) 包含两项: **/.:/eng/obj1** 和 **/.:/eng/link1**。目标目录 (**/.:/rnd**) 也包含两项: **/.:/rnd/obj2** 和 **/.:/rnd/link2**。

合并 CDS 目录

以下过程将源目录 **/.:/eng** 合并到目标目录 **/.:/rnd** 中:

1. 将 **/.:/eng** 目录合并到 **/.:/rnd** 目录之前，先藏匿该目录。源目录副本的这个同步过程能够防止导致合并操作失败的错误。

   ```
   dcecp> directory synchronize /.:/eng
   dcecp>
   ```

2. 运行 **directory merge** 命令来合并 **/.:/eng** 和 **/.:/rnd** 目录:

   ```
   dcecp> directory merge /.:/eng -into  /.:/rnd
   dcecp>
   ```

   请注意，**directory merge** 命令只合并命令行变量中命名的源目录的直接内容（即，这些目录中的对象项、软链接和子目录）。

   要将目录中任何子目录的子代复制到目标位置，必须使用命令的 **-tree** 选项。例如，如果上例中的 **/.:/eng** 目录包括子目录 **dev** 和 **qa**，并且要将这些目录的内容合并到目标目录 **/.:/rnd** 中，应输入以下命令行:

   ```
   dcecp> directory merge /.:/eng -into /.:/rnd -tree
   dcecp>
   ```

   缺省情况下，**directory merge** 命令将所有对象项、软链接和子目录放到目标目录的主中央交换库中。但是，可以将子目录放到另一个中央交换库中。要这样做，应使用命令的 **-clearinghouse** 选项来指定其它中央交换库的名称。

   请注意，只允许在 **-clearinghouse** 选项中指定一个备用中央交换库。如果希望将子目录放到不同的备用中央交换库中，必须为每个中央交换库发出单独的 **directory merge** 命令，或者必须发出单一 **directory merge** 命令将所有子目录放到一个中央交换库中，然后在合并操作后重新定位目录。

   注: **directory merge** 命令创建的 CDS 对象保留源对象的所有可写属性值和一些只读属性值。但是，这些对象不继承源对象的 ACL。如果合并的对象是目录，那么 **directory merge** 命令将给予它初始容器的缺省 ACL。如果合并的对象是其它任何 CDS 类型，**directory merge** 命令将给予它初始对象的缺省 ACL。

   如果 **directory merge** 命令在合并操作中遇到问题，它有两种处理方式。如果包括 **-nocheck** 选项，那么命令在执行操作之前不会检查错误。它将直接执行操作，如果遇到错误，则停止。如果省略 **-nocheck** 选项，那么命令在开始合并之前将检查某些错误条件。如果找到错误，它将显示错误消息并停止；否则，继续合并。

   directory merge 命令返回的错误消息指出引起问题的 CDS 实体，并提供问题的简要描述。再次运行命令之前，应解决命令遇到的问题。（有关在合并操作期间可能发生的错误类型的详细信息，请参阅第185页的『处理错误』。）

3. 合并操作后，**/.:/eng** 目录（及其内容）仍在源位置中。运行以下命令，从它原始位置中删除 **/.:/eng** 目录，并创建软链接 **/.:/eng** 来代替所删除的目录。软链接将把 **obj1** 和 **link1** 对象项的查找重定向到它们在 **/.:/rnd** 目录中的新位置。

   建议在删除它之前，先藏匿源目录。目录副本的这个同步过程能够防止导致删除操作失败的错误。

   以下一系列命令用于同步和删除 **/.:/eng** 目录，然后创建以前内容的软链接:

```
dcecp> directory synchronize  /.:/eng
dcecp> directory delete  /.:/eng  -tree
dcecp> link create  /.:/eng  -to  /.:/rnd
dcecp>
```

调用 **-tree** 选项的 **directory delete** 命令将删除一个目录，以及该目录下面的所有对象项、软链接和子目录。如果使用 **directory delete** 命令但不选用 **-tree** 选项，必须清空所有要删除的目录，否则会出错。

图20 显示了本例中合并操作前后的示例名称空间的结构。

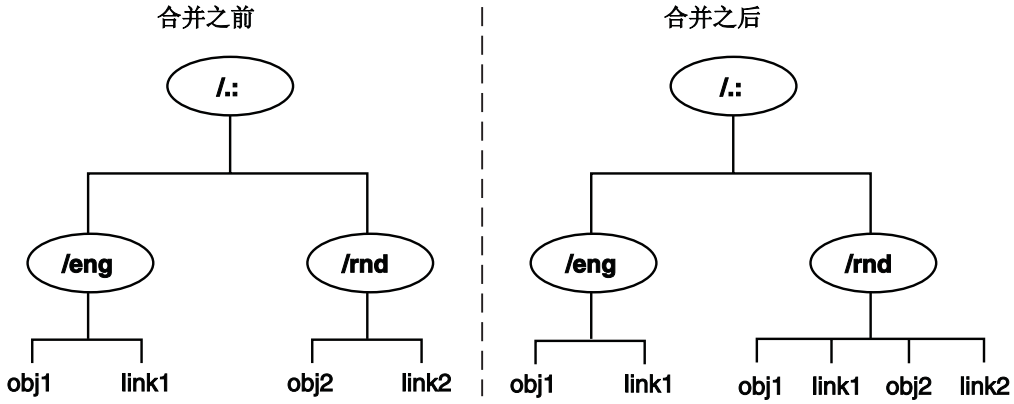


图 20. 合并操作之前和之后的示例名称空间

附加 CDS 目录

以下过程将源目录 **/.:/eng** 附加到 **/.:/rnd** 目录（即，将 **/.:/eng** 目录复制到 **/.:/rnd** 目录下空的目标目录 **/eng** 中）：

1. 运行 **directory create** 命令来创建新的空目录 **/.:/rnd/eng**，然后可以将源目录 **/.:/eng** 的内容放到该目录中:

   ```
   dcecp> directory create /.:/rnd/eng
   dcecp>
   ```

   缺省情况下，**directory create** 命令将新目录创建在与父目录相同的中央交换库中。如果希望将目录创建到另一个中央交换库中，必须使用命令的 **-clearinghouse** 选项来指定其它中央交换库。

2. 将 **/.:/eng** 目录附加到 **/.:/rnd** 目录之前，先藏匿它。源目录副本的这个同步过程能够防止导致附加操作失败的错误:

   ```
   dcecp> directory synchronize /.:/eng
   dcecp>
   ```

3. 运行 **directory merge** 命令将源目录 **/.:/eng** 附加到 **/.:/rnd** 目录（或将它合并到新的 **/.:/rnd/eng** 目录中）：

   ```
   dcecp> directory merge /.:/eng -into /.:/rnd/eng
   dcecp>
   ```

   如果源目录包含任何要复制其内容的子目录，则必须在 **directory merge** 命令行中指定 **-tree** 选项。另外，如果希望将子目录及其内容从 **/.:/rnd/eng** 目录放到另一个中央交换库中，需要指定 **-clearinghouse** 选项。

如果合并操作没有成功，可以删除目标位置中部分合并的信息，并再次运行命令。可是，要确保删除所有重复的名称，并确定可以维护与受影响的中央交换库的连接。

注: directory merge 命令创建的 CDS 对象保留源对象的所有可写属性值和一些只读属性值。但是，这些对象不继承源对象的 ACL。目标对象的 ACL 不是从初始容器（对象合并到的父目录）继承的 ACL 就是从初始对象继承的 ACL。

4. 附加操作后，**/.:/eng** 目录（及其内容）仍在源位置中。需要从它原始位置中删除 **/.:/eng** 目录，并创建软链接 **/.:/eng** 来代替所删除的目录。软链接将把 **obj1** 和 **link1** 对象项的查找重定向到它们在 **/.:/eng** 目录中的新位置。

   建议在删除它之前，先藏匿源目录。目录副本的这个同步过程能够防止导致删除操作失败的错误。

   以下一系列 **dcecp** 命令用于从源位置中除去 **/.:/eng** 目录:

```
dcecp> directory synchronize /.:/eng
dcecp> directory delete /.:/eng
dcecp> link create /.:/eng -to /.:/rnd/eng
dcecp>
```

   图21 显示了附加操作前后的示例名称空间的结构。

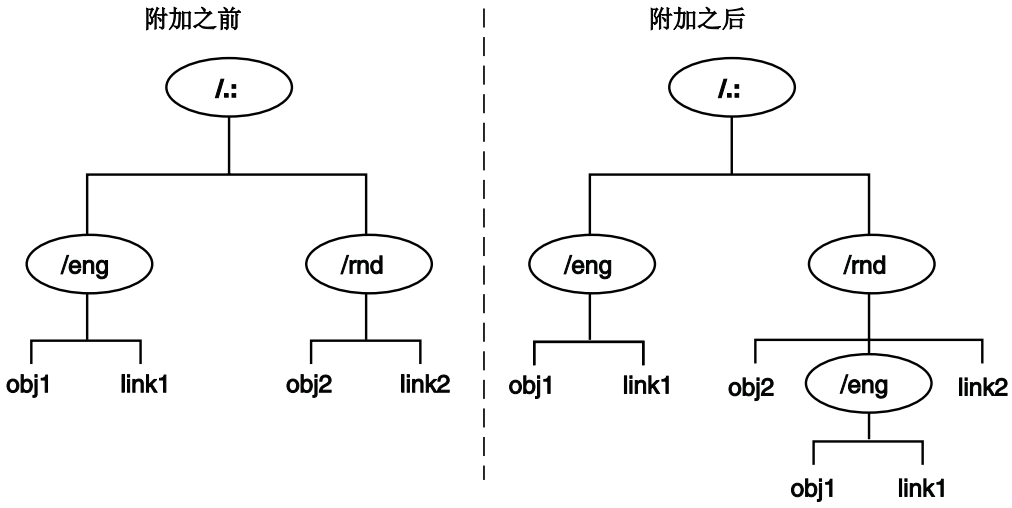


图 21. 附加操作之前和之后的示例名称空间

修改目标位置上的 ACL

要将主体对合并信息的访问权保存到目标目录中，在目标位置中新创建的对象的 ACL 需要与源目录中对象的 ACL 匹配。因为 **directory merge** 命令不重新创建新位置处 CDS 对象的源 ACL，您可能需要在合并操作后修改目标 ACL。要修改这些 ACL，使用 **dcecp acl replace** 或 **acl modify** 命令，这取决于要替换整个 ACL 还是只想修改 ACL 项。

处理错误

directory merge 命令在操作期间遇到的大多数错误都是由以下原因引起的:

- 合并期间检测到的重名
- 名称在源子树中，该子树的主中央交换库在命令执行时不可访问
- 由于许可权不足而没有在目标位置中创建项

以下部分说明了如何从这些错误中恢复。

重名

如果 CDS 对象项或软链接的全名与目标位置上的对象项或软链接的全名相同，**directory merge** 命令会列出这些重复名称，然后停止。不会合并重复名称，以避免覆盖和毁坏目标目录中的相同名称。

如果存在重复的名称，则需要决定要保留哪些名称: 源子树中的名称还是目标子树中的名称。一旦做出决定，以下列方式执行:

1. 使用 **dcecp create** 操作在源子树或目标子树中重新创建（用新的名称）任何重复对象项或软链接，使它们成为新的对象项或软链接。然后删除重复名称。
2. 当确定可以维护受影响中央交换库的连接时，重新运行 **directory merge** 命令来合并源目录和目标目录的内容。

不可访问名称故障

有时，在输入 **directory merge** 命令时，存储想要合并的目录主副本的中央交换库被禁用或不可访问。发生这种情况时，该命令不能在新的目标位置中创建目录及其包含的项。

如果由于这个原因无法合并名称，**directory merge** 命令将显示一条错误消息，指出不能创建的名称，然后终止。

许可权不足

如果缺少适当的许可权，**directory merge** 命令就不能在目标位置创建 CDS 对象。如果命令返回错误消息表示许可权不足，则需要检查目标中央交换库、目录和对象项的 ACL 来查看当前的许可权，并更改错误的许可权。

表12 显示了在目标位置创建目录和其它 CDS 对象项所需的许可权。

表 12. 创建目标对象必需的许可权

对象	必需的许可权
目录	对将存储新目录主副本的中央交换库的写许可权。 对新目录的父目录的插入和读许可权。 对新目录的初始容器的插入和读许可权。 服务器主体还需要对新目录的父目录的读和插入许可权。
其它 CDS 对象	要在其中创建对象的目录的插入和读许可权。该对象类型的初始对象的插入和读许可权。

将 CDS 目录合并到外部单元中

还可以使用 **directory merge** 命令将 CDS 目录合并到外部单元的名称空间中。通常，执行的过程与用于合并相同名称空间中的目录或子树的过程相同。但是，需要记住一些附加注意事项:

- 需要提前建立单元间的认证。
- 需要合并源单元和目标单元中的整个目录层次结构。

另外，在同一个名称空间中合并目录时，需要修改新建目标对象的 ACL。

建立单元间认证

如果要使源单元中的用户和应用程序可以方便地继续访问目标单元中的合并信息，请确保在源单元和外部（目标）单元之间有单元间认证的协议。否则，就不允许源单元中请求新合并信息的主体与目标单元通信。有关如何建立单元间认证的完整信息，请参阅第249页的『第6部分 DCE 安全性服务』。

执行合并到外部单元的操作

要将 CDS 数据合并到外部单元的名称空间中，执行以下这些步骤:

1. 当登录到外部单元中目标机器上的特权帐户（**cell_admin** 或 **cds-admin** 组成员）时，运行 **directory merge** 命令将源单元的目录的内容与现有目录合并。
2. 如果要继续访问源单元中的合并信息，删除源子树中的最高层目录，并用与目录同名的软链接替换已删除的信息。这将把信息查找重定向到外部单元中的新位置。

恢复合并的 CDS 目录

可以使用 **dcecp link delete** 和 **directory merge** 命令将已删除的目录及其内容恢复到名称空间中。

首先，运行 **link delete** 命令来除去以前源位置中的软链接，然后使用 **directory merge** 命令将目录的副本附加回它以前的父目录下。

如果目录有只读副本，使用 **directory create** 命令在每个已从中删除目录的中央交换库中创建目录的新副本。

请记住，**directory merge** 命令只影响目录及其内容。它不会复制中央交换库或关联的中央交换库对象项，因此不能用于恢复中央交换库或解决驻留在不同的中央交换库上的个别副本之间的信息差异。而且，自从原始合并操作之后，特定位置中的目录信息可能已经更改。

第21章 重构名称空间

随着时间的过去，可能需要重构或重命名名称空间的某些元素。例如，可能想要创建软链接，让用户可以使用现有名称空间项的一个或多个备用名称。可能要重新配置目录的副本集，以修改特定副本的位置和副本类型，或者从集合中排除某个副本。有时候，当用户不再需要某些目录中的信息时，可能想要删除这些目录。也可能需要重新定位中央交换库或者从服务器系统中删除某个中央交换库，以便在系统中执行诊断或故障排除，或者准备从网络中除去系统。最后，您可能想要创建单元层次结构，将单元添加到现有层次结构，或者更改单元层次结构的构成。

本章说明了如何执行以下名称空间重构任务:

- 管理软链接
- 修改目录的副本集
- 删除目录
- 重新定位中央交换库
- 删除中央交换库

管理软链接

软链接是一个备用名称或别名，可以使用它来引用名称空间中另一个现有名称。软链接允许用户和客户机应用程序使用多个名称来引用特定目录、对象项或软链接。

通常，应该创建一个软链接将备用名称分配给特定网络资源，或者对名称空间层次结构中的原始目录名稍做更改。应该避免使用软链接来完全重新设计名称空间。

创建软链接

使用 DCE 控制程序 (**dcecp**) **link create** 命令来创建软链接。除了新的软链接名称，还必须使用 **-to** 选项来指定软链接的目的地名称，或者新软链接指向的现有名称。可以将本地单元名称空间或任何外部单元名称空间中的任何名称指定成目的地名称，包括另一个软链接。

要创建软链接，必须对要在其中创建软链接的目录有插入许可权。

注: 如果创建一个软链接，而它又指向另一个软链接，请确保没有创建软链接循环。如果指定的目的地名称最终又指向新链接自己的链接名称，那么这就是软链接循环。职员程序可以检测出这种错误。

使用 **link create** 命令创建的所有软链接都是永久性的，永远不会失效，除非使用命令的 **-timeout** 选项来指定失效日期和软链接的 **CDS_LinkTimeout** 属性的时间值。

以下列格式输入失效日期和时间值:

yyyy-mm-dd-hh:mm:ss

例如，下列值表示如果软链接在 2010 年 8 月 25 日下午 4 点仍然存在（即，尚未被手工被删除），CDS 将在下一次藏匿存储它的目录时自动删除它:

```
CDS_LinkTimeout=(2010-08-25-16:00:00)
```

如果使用 **-timeout** 选项来指定软链接的 **CDS_LinkTimeout** 属性的失效值，还可以指定一个延期值，这个值是要添加到已指定失效日期和时间的一段时间。以 *ddd-hh:mm:ss* 格式输入延期值。例如，**030-00:00:00** 值表示如果到了指定的失效日期和时间，软链接的目的地名称仍然存在，CDS 在藏匿期间允许在下次检查之前目的地名称再存在 30 天。到那时，如果找不到目的地名称，那么 CDS 将删除软链接。

以下命令创建了一个永久软链接 **/.:/sales/asia**，它指向目录 **/.:/sales/eur**:

```
dcecp> link create /.:/sales/asia -to /.:/sales/eur
dcecp>
```

以下命令创建了软链接 **/.:/mfg/robo1**，它指向对象项 **/.:/mfg/robotics_controller01**，并设置了失效日期和时间:

```
dcecp> link create /.:/mfg/robo1 -to /.:/mfg/robotics_controller01 \
 -timeout 2010-12-12-09:00:00
dcecp>
```

以上命令中，**CDS_LinkTimeout** 属性值中的失效日期和时间表示 CDS 将在 2010 年 12 月 12 日上午 9 点之后的第一次藏匿时删除软链接 **/.:/mfg/robo1**。

以下命令创建了软链接 **/.:/admin/linka**，它指向对象项 **/.:/sales/discount_stats**:

```
dcecp> link create  /.:/admin/linka -to /.:/sales/discount_stats -timeout \
 {2010-01-11-12:00:00 090-00:00:00}
dcecp>
```

以上命令中，**CDS_LinkTimeout** 属性值中的失效时间表示 CDS 将在 2010 年 1 月 11 日午夜 12 点之后的第一次藏匿时检查目的地名称 **/.:/sales/discount_stats** 是否还存在。如果目的地名称不存在，那么 CDS 将删除软链接。如果目的地名称仍然存在，那么按 **CDS_LinkTimeout** 属性值 **090-00:00:00** 指定的延期时间，将软链接再保留 90 天。当 90 天延期时间失效时，CDS 会按 90 天的间隔重复检查，直到目的地名称被删除为止。

更改软链接的目的地名称

使用 **dcecp link modify** 命令为软链接的 **CDS_LinkTarget** 属性指定新值，并将软链接从当前目的地名称重定向到名称空间中的其它名称。

要更改软链接的目的地名称，必须拥有软链接的写许可权。例如，以下命令将软链接 **/.:/admin/work_disk** 从当前目的地名称重定向到新目的地名称 **/.:/admin/work_disk03**:

```
dcecp> link modify  /.:/admin/work_disk -change {CDS_LinkTarget \
 /.:/admin/work_disk03}
dcecp>
```

更改软链接的失效或延期值

使用 **dcecp link modify** 命令为存储在软链接的 **CDS_LinkTimeout** 属性中的失效和延期值指定新值。即使只想修改一个值，但在命令中还必须全部指定失效和延期值。指定新值时用的格式应该与建立原始值的格式相同。失效值的格式是 *yyyy-mm-dd-hh:mm:ss*，延期值的格式是 *ddd-hh:mm:ss*。

要更改软链接的失效和延期值，必须有该软链接的写许可权。

以下命令将软链接 **/.:/eng/link01** 的失效值设置为 2010 年 12 月 31 日午夜 12 点。在本例中，当前没有对软链接指定延期时间。

```
dcecp> link modify /.:/eng/link01 -change {CDS_LinkTimeout \
 (2010-12-31-12:00:00 000-00:00:00}
dcecp>
```

以下命令将软链接 **/.:/eng/link01** 的失效值更改成 2010 年 12 月 31 日午夜 12 点，并将软链接的延期值设置成 90 天:

```
dcecp> link modify /.:/eng/link01 -change {CDS_LinkTimeout \
 2010-12-31-12:00:00 090-00:00:00}
dcecp>
```

删除软链接

如果发现永久软链接已经超出了其原来的预期寿命，或者不想等到软链接的指定失效或延期时间，那么您可以自己从名称空间中删除该软链接。

使用 **link delete** 命令删除指定名称的软链接。

要删除软链接，必须拥有该软链接的删除许可权，或者存储软链接的目录的管理许可权。

例如，以下命令删除软链接 **/.:/dist/pointer_1**:

```
dcecp> link delete  /.:/dist/pointer_1
dcecp>
```

修改目录的副本集

目录的副本集始终包含主副本；它还包含其它只读副本。存储在与目录相关的 **CDS_Replicas** 属性中的值包含与目录的副本集相关的信息，包括存在多少副本、它们的副本类型、以及存储每个副本的中央交换库的名称。可以使用 **dcescp directory modify** 命令来覆盖存储在目录的 **CDS_Replicas** 属性中的值，并执行单一命令来完成以下一个或全部的任务:

- 在目录的副本集中指定新的主副本。
- 从目录的副本集排除副本。

注: 作为 **directory modify** 命令功能的一部分，CDS 会立即藏匿目录，以便尽快将修改分布到副本集的所有成员。

修改副本集之前

在修改目录的副本集之前，需要知道存在多少副本、它们的副本类型、以及存储每个副本的中央交换库的名称。用于修改目录副本集的命令不允许将副本遗漏在新的集合之外。必须明确列出该集合中的全部现有副本。可以将任何副本包括到新集合中，或者从新集合中排除任何副本，但必须考虑到所有副本。在所有包括在新集合中的副本中，只有一个可以指定为主副本。

要显示所有目录副本的名称，使用 **dcecp directory show** 命令。该命令将查询目录的 **CDS_Replicas** 属性以搜集该信息。（有关如何使用 **dcecp directory show** 命令的信息，请参阅第169页的『第18章 管理 CDS 目录』。）

修改副本集必需的许可权

修改目录的副本集的许可权如下:

- 必须拥有目录的管理许可权。另外，服务器主体需要目录的管理、读和写许可权。
- 当指定新的主副本时，还需要有存储当前主副本的中央交换库的写许可权。如果要将只读副本指定为新的主副本，那么服务器主体需要有存储该只读副本的中央交换库的写许可权。

  将存储新主副本的服务器上的服务器主体需要该目录的管理、读和写许可权。

当知道了要包括和排除哪些副本并已经更改了需要更改的许可权，发出 **directory modify** 命令来修改目录的副本集。以下部分给出了两个选项--指定新的主副本，排除现有只读副本--的指令。

指定新的主副本

有时，由于配置管理方面的原因，可能想要指定另一个副本作为目录的主副本。

例如，在以下情况下，可以指定新的主副本:

- 其中央交换库包含了一个或多个主副本的服务器系统将停机一段时间，或者永远从网络中除去。
- 将从名称空间中删除存储了一个或多个主副本的中央交换库。
- 想要使主副本更接近发生多数目录更新的位置。

要指定新的主副本，使用 **dcecp directory modify** 命令。

图22 显示了示例副本集。 **/.:/eng** 目录的这个副本集由三个副本组成: 存储在中央交换库 **/.:/NY1_CH** 中的主副本，存储在中央交换库 **/.:/NY2_CH** 中的只读副本，以及存储在中央交换库 **/.:/Chicago1_CH** 中的只读副本。

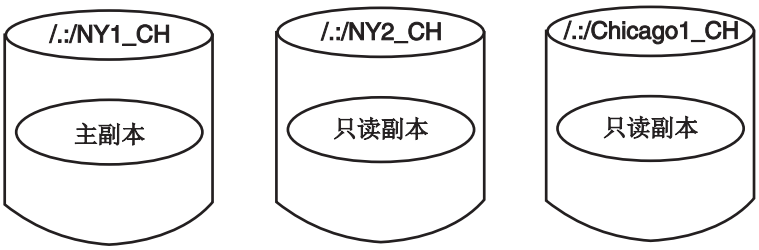


图 22. 示例副本集

以下命令将存储在中央交换库 **/.:/Chicago1_CH** 中的只读副本指定成主副本，将以前的主副本（存储在中央交换库 **/.:/NY1_CH** 中）指定成只读副本，并使存储在中央交换库 **/.:/NY2_CH** 中的只读副本保留原样:

```
dcecp> directory modify /.:/eng -master /.:/Chicago_1_CH \
> -readonly {/.:/NY1_CH /.:/NY2_CH}
dcecp>
```

第191页的图23 显示了以上命令的结果。

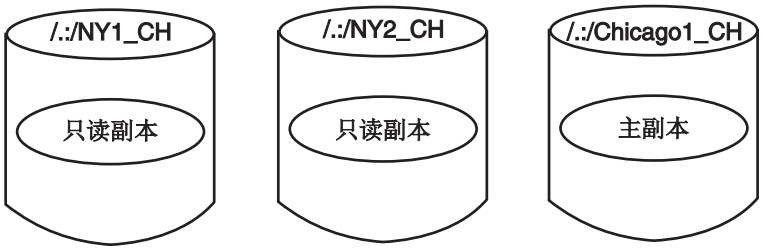


图 23. 重新指定主副本之后的示例副本集

从副本集排除副本

当存储副本的中央交换库意外地变成不可用时，可以临时从副本集排除该副本。在排除的副本不可用期间，这样做可以使 CDS 能够藏匿目录。

要从副本集排除副本，使用 **dcecp directory modify** 命令，并选用 **exclude** 变量来重新构建目录的副本集，指定的副本除外。请记住，在命令中必须考虑到所有现有副本。

下例中，**/.:/eng** 目录的副本集由三个副本组成：存储在中央交换库 **/.:/Chicago1_CH** 中的主副本，和存储在中央交换库 **/.:/NY1_CH** 和 **/.:/NY2_CH** 中的只读副本。

这种情况中，**/.:/NY1_CH** 中央交换库会从网络中断开，否则会对网络传输线路产生意外的损坏。与中央交换库的连通性在几天之内不会恢复。在这段时间内，藏匿 **/.:/eng** 目录将会失败，除非临时排除存储在中央交换库 **/.:/NY1_CH** 中的只读副本。

要使藏匿 **/.:/eng** 目录可能在修复期内成功，输入以下命令来用只包括存储在 **/.:/NY2_CH** 和 **/.:/Chicago1_CH** 中央交换库中的副本的新值覆盖 **/.:/eng** 目录 **CDS_Replicas** CDS 的当前值:

```
dcecp> directory modify/.:/eng -master /.:/Chicago1_CH \
> -readonly /.:/NY2_CH -exclude /.:/NY1_CH
dcecp>
```

图24 显示了以上命令的结果。

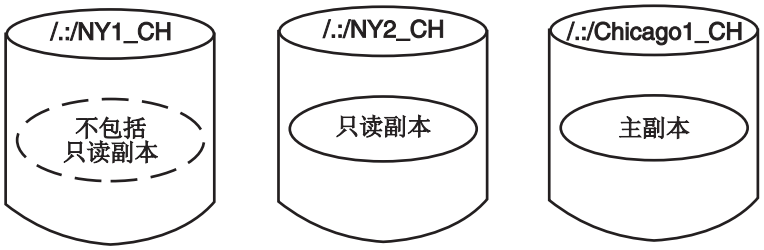


图 24. 副本排除之后的示例副本集

当重新建立与 **/.:/NY1_CH** 中央交换库的连通性时，输入以下命令将存储在中央交换库 **/.:/NY1_CH** 中的只读副本引入只读副本:

```
dcecp> directory modify /.:/eng -master /.:/Chicago1_CH \
> -readonly {/.:/NY1_CH /.:/NY2_CH}
dcecp>
```

注: 当排除的副本驻留的中央交换库再次变成可用之后，应尽快将排除的副本重新引入副本集。

删除目录

当用户不再需要某个目录包含的信息时，可以从名称空间中删除该目录。当删除目录时，必须考虑两点:

- 目录是否包含子目录或其它任何 CDS 对象项？可以删除目录之前，它必须是空的。
- 目录有副本吗？必须单独删除每个副本。

以下部分讨论了这些注意事项。

要删除目录，必须拥有下列许可权:

- 目录的删除许可权。
- 对存储目录主副本的中央交换库的写许可权。
- 在其中输入 **directory delete** 命令的服务器的服务器主体需要有父目录的管理许可权，或指向要删除目录的子指针的删除许可权。

  如果该服务器包括在服务器授权组 **subsys/dce/cds-servers** 中，这些许可权应该已经就绪。如有怀疑，使用 **dcecp** 实用程序的 **acl show**，并验证服务器主体是否具有相应的许可权。(有关 **acl show** 命令的完整信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference*。)

删除无副本的目录

要删除无副本的目录，使用 **dcecp directory delete** 命令。例如，要删除目录 **/.:/sales**、它的所有直接内容及其所有子目录的内容，应输入以下命令:

```
dcecp> directory delete  /.:/sales -tree
dcecp>
```

注: 使用 **directory delete** 命令的 -tree 时请小心。命令不会要求您确认想要删除在命令中指定的目录；它会立即执行删除操作。这会导致丢失要保留的目录。

请记住，可以通过脚本来更改 **dcecp** 命令的行为。例如 **directory delete** 命令，可以编写一个脚本，每当使用命令时指定 **-tree** 选项，它都会提示确认删除操作。有关编写脚本的讨论，请参阅本指南的第1页的『第1部分 DCE 控制程序』。

一种防止无意删除目录及其项的方法是在运行 **directory delete** 命令之前先查看内容。要按项类型显示 CDS 目录的内容，使用 **directory list** 命令并选用 **-object**、**-link** 和 **-directory** 选项。

下例中显示了删除目录 **/.:/sales**。目录有一个对象项和软链接:

```
dcecp> directory list /.:/sales -simplename work_disk link1
dcecp> directory list /.:/sales -simplename -object work_disk
dcecp> directory list /.:/sales -simplename -link link1
dcecp> directory delete /.:/sales -tree
dcecp> directory show /.:/sales
错误: 所请求的项不存在
dcecp>
```

如果要删除的目录不是空的，那么 **directory delete** 命令将失败。要避免这种失败，必须除去目录及其子目录中的所有项，然后再次运行 **directory delete** 命令。使用 **link delete** 和 **object delete** 命令来删除任何目录中的软链接和对象项。然后运行 **directory delete** 命令来删除目录。

删除目录副本

如果要复制目录，那么必须单独删除所有副本。然后可以使用以上部分中描述的命令来删除目录。

要显示目录所有副本的列表，使用 **dcecp directory show** 命令。查看列表中目录的 **CDS_Replicas** 属性。每个副本的 **CDS_Replicas** 属性都有几个子属性。查看每个副本的 **CH_Name** 子属性来获取它所在的中央交换库的名称。例如:

```
dcecp> directory show /.:/sales
{RPC_ClassVersion {01 00}}
{CDS_CTS 2010-05-06-11:41:05.314-05:00I0.000/08-00-09-25-13-52}
{CDS_UTS 2010-06-21-03:06:08.842-05:00I0.000/08-00-09-25-13-52}
{CDS_ObjectUUID 5f97a584-bf9b-11cd-9362-080009251352}
{CDS_Replicas
 {{CH_UUID de3401e6-bb98-11cd-aac5-080009251352}
  {CH_Name /.../absolut_cell/absolut_ch}
  {Replica_Type Master}
  {Tower {ncacn_ip_tcp 130.105.5.93}}
  {Tower {ncadg_ip_udp 130.105.5.93}}}}
{CDS_AllUpTo 23854-01-29-19:45:44.841-05:00I0.000/08-00-09-25-13-52}
{CDS_Convergence medium}
{CDS_ParentPointer
 {{Parent_UUID df13b228-bb98-11cd-aac5-080009251352}
  {Timeout
   {expiration 2010-08-24-19:30:30.827}
   {extension +1-00:00:00.000I0.000}}
  {myname /.../absolut_cell/sales}}}
{CDS_DirectoryVersion 3.0}
{CDS_ReplicaState on}
{CDS_ReplicaType Master}
{CDS_LastSkulk 2010-01-29-19:45:44.841-05:00I0.000/08-00-09-25-13-52}
{CDS_LastUpdate 2010-06-21-03:06:08.842-05:00I0.000/08-00-09-25-13-52}
{CDS_Epoch 60ac0730-bf9b-11cd-9362-080009251352}
{CDS_ReplicaVersion 3.0}
dcecp>
```

目录名和中央交换库的名称可以用来唯一标识每个副本。在一系列 **directory delete** 命令中使用这些名称来除去副本。每个副本的名称都是命令的变量，中央交换库的名称应用作 **-clearinghouse** 选项的值。命令行中还应出现 **-replica** 选项以表示要删除的目录是副本。样本命令行如下:

```
dcecp> directory delete /.:/sales -replica -clearinghouse /.:/NY1_CH
dcecp>
```

注: **directory delete** 命令不要求目录是空的才可以删除。它会立即删除副本、它们的所有内容和子目录，而不提示确认操作。

您可能想要编写一个 **dcecp** 脚本，这样可以用一个命令来查看 **CDS_Replicas** 属性，查找所有副本并删除它们。有关编写脚本的讨论，请参阅本指南的第1页的『第1部分 DCE 控制程序』。

重新定位中央交换库

注: 本节描述用于临时将中央交换库从一个 CDS 服务器重新定位到另一个 CDS 服务器的过程。请注意，该过程不能用于配置附加 CDS 服务器系统。（有关如何配置 CDS 服务器和 CDS 职员程序的信息，请参阅 *IBM DCE 版本 3.2 AIX 和 Solaris 版: 管理指南--简介*。）

有时，可能需要将中央交换库从其当前驻留的服务器系统重新定位到另一个服务器系统。例如，在以下情况中，可能要移动中央交换库:

- 由于要修理或其它原因，需要临时将主机服务器系统从网络中断开。
- 不再希望当前主机系统充当 CDS 服务器。
- 想要将中央交换库移到网络上的一个服务器系统，该系统在物理位置上更接近使用中央交换库中包含的信息的用户组和应用程序。

要重新定位中央交换库，执行以下步骤:

1. 以 cell_admin 登录到 CDS 服务器 A 和 B。
2. 在服务器 A 上输入以下命令。

   ```
   dcecp>  dcecp -c clearinghouse
   disable /.:/my_ch
   ```

3. 将服务器 A 上的目录更改成 **/opt/dcelocal/var/directory/cds**。
4. 检查 **/.:/my_ch** 相关项的 **cdsfiles.map** 文件。它们看上去如下:

   ```
   /opt/dcelocal/var/directory/cds/acell#my_ch.checkpoint001
   /opt/dcelocal/var/directory/cds/acell#my_ch.tlog001
   /opt/dcelocal/var/directory/cds/acell#my_ch.version
   ```

5. 将步骤 4 中找到的文件传送到服务器 B 上的相同目录。
6. 从服务器 A 的 **cdsfiles.map** 文件中抽取在步骤 4 中找到的映射项，并添加到服务器 B 的 **cdsfiles.map** 文件。
7. 在服务器 A 上，删除服务器 A 的 **cdsfiles.map** 文件中在步骤 4 中找到的项以及与它们关联的文件。
8. 在服务器 B 上输入以下命令:

   ```
   dcecp>  dcecp -c clearinghouse create /.:/my_ch
   ```

9. 在服务器 B 上输入以下命令:

   ```
   dcecp>  dcecp -c cdscache discard
   ```

将中央交换库从其主机服务器系统中分离

每当 CDS 服务器启动时，服务器软件执行的任务之一就是启动它的中央交换库（或多个中央交换库）。服务器通过检查驻留在该系统上的中央交换库的列表来自动执行该任务。在重新定位中央交换库之前，使用 **dcecp clearinghouse disable** 命令来更新中央交换库文件，并且确保在将这些文件复制到目标服务器之前，它们是一致的。

clearinghouse disable 命令还从源服务器的内部内存中除去所指定的中央交换库的知识。这确保了在服务器重新启动期间，不会在源服务器上自动启动重新定位的中央交换库。

要使用 **clearinghouse disable** 命令，必须对中央交换库驻留的服务器拥有写许可权。

以下示例命令从主机服务器的内存中除去中央交换库 **/.:/Chicago2_CH** 的知识:

```
dcecp> clearinghouse disable /.:/Chicago2_CH
dcecp>
```

将中央交换库数据库文件复制到目标服务器系统

禁用中央交换库并从主机服务器上除去中央交换库的知识之后，必须将中央交换库数据库文件复制到新主机服务器上的特定位置。

中央交换库数据库由以下三个文件组成:

- *clearinghouse-name*.**checkpoint***nnnnnnnn*
- *clearinghouse-name*.**tlog***nnnnnnnn*
- *clearinghouse-name*.**version**

其中 *nnnnnnnn* 表示中央交换库名称。

在试图将这些文件复制到新主机系统之前，应该验证它们是否存在。（有关所有 CDS 文件的完整路径名，请参阅 *IBM DCE 版本 3.2 AIX 和 Solaris 版: 管理指南--简介*和 *OSF DCE Porting and Testing Guide*。）

注: 有时，可以在目录中找到两个 **.checkpoint***nnnnnnnn* 文件。在中央交换库的最新检入操作期间，这可能会引起系统崩溃或其它中断。如果确实找到两个文件，将它们都复制到目标服务器系统。在目标服务器上启用了中央交换库之后，该系统上的服务器软件会尽快自动调节可能存在的问题。

要将数据库文件移到新的 CDS 服务器，使用 **ftp** 实用程序或类似的网络文件传送实用程序。将三个数据库文件从现有服务器主机复制到新的 CDS 服务器主机。在旧的和新的 CDS 服务器上，文件驻留的目录是 *dcelocal***/var/directory/cds**。

启动目标服务器上的中央交换库

将中央交换库数据库文件复制到目标服务器系统上的适当位置之后，使用 **clearinghouse create** 命令在新的位置上启动中央交换库。请确保指定的中央交换库名称与其原始（源）位置上使用的名称相同。命令输入后，服务器将检测中央交换库文件，将它们的知识添加到它的内存中，然后启动中央交换库。

要使用 **clearinghouse create** 命令来重新定位中央交换库，必须有要重新定位中央交换库的服务器的写许可权。

上例中，成功将中央交换库 **/.:/Chicago2_CH** 的数据库文件复制到服务器系统 **orion**。在 **orion** 上发出的下列命令在该服务器上重新定位中央交换库 **/.:/Chicago2_CH**:

```
dcecp> clearinghouse create /.:/Chicago2_CH
dcecp>
```

删除中央交换库

发生以下情况时，可能需要从中央交换库驻留的服务器中删除该中央交换库:

- 已计划重新分配或从网络中除去该系统。
- 不再希望系统充当 CDS 服务器。

删除中央交换库之前

在试图删除中央交换库之前，请确保:

- 中央交换库对服务器是已知的。
- 中央交换库没有存储主副本。

清除中央交换库时，运行中央交换库的服务器的内部内存中不再拥有关于该中央交换库的知识。如果以后尝试删除中央交换库，CDS 将找不到它，并返回消息指出它不存在。在删除已清除中央交换库之前，必须使用 **clearinghouse create** 命令重新创建它。

CDS 不允许删除包含目录主副本的中央交换库。删除这种中央交换库之前，必须指定该目录副本集的另一个副本作为主副本。如果目录没有其它副本，在另一个中央交换库中创建一个只读副本，然后在从中央交换库中删除原始主副本之前，将它指定为目录新的主副本。（有关修改目录副本集的指令，请参阅第189页的『修改目录的副本集』。）

删除中央交换库的许可权

删除中央交换库需要以下许可权:

- 需要有中央交换库的写和删除许可权，以及将副本存储在该中央交换库中的所有目录的管理许可权。
- 服务器主体需要对相关中央交换库对象项的删除许可权，以及对将副本存储在该中央交换库中的所有目录的管理许可权。

删除中央交换库

使用 **clearinghouse delete** 命令来删除中央交换库。该命令还删除中央交换库的关联中央交换库对象项，以及中央交换库中的所有只读副本。

完成中央交换库删除可能需要一定时间。只有在成功完成了将只读副本存储在中央交换库中的所有目录藏匿后，CDS 才删除中央交换库。

以下示例命令删除 **/.:/Paris2_CH** 中央交换库:

```
dcecp> clearinghouse delete /.:/Paris2_CH
dcecp>
```

第22章 管理单元间命名

要查找本地单元之外的名称，CDS 职员程序必须有办法可以查找其它单元的目录服务器。全局目录代理 (GDA) 通过充当全局命名环境中与其它单元的连接来启用单元间通信。本章描述了 GDA 如何工作以及如何管理它。本章还描述了如何在（X.500、LDAP 或 DNS）中定义本地单元，其中有一步对于使本地单元可以被其它单元访问至关重要。

注: 如果单元名称是 X.500 形式的名称，那么 GDS 或 LDAP 服务器就可以用作全局名称服务器。

全局目录代理的工作原理

GDA 是本地单元中 CDS 职员程序和其它单元中 CDS 服务器之间的中间件。 CDS 职员程序对待 GDA 就像其它 CDS 服务器一样，将名称查找请求传递给它。然而，GDA 只向职员程序提供一项特别服务：它在 X.500、LDAP 或 DNS 名称空间中查找单元名称，并将结果返回给职员程序。然后职员程序使用那些结果来联系外部单元中的 CDS 服务器。

任何想要与其它单元通信的单元都必须有 GDA。GDA 可以与 CDS 服务器在同一个系统上，或者它可以独立存在于另一个系统中。可以在一个单元中配置多个 GDA 以增加可用性和可靠性。就像 CDS 服务器一样，GDA 是一个主体，并将向职员程序认证它自己。

CDS 通过读取存储在与单元根目录相关的 **CDS_GDAPointers** 属性中的地址信息来查找 GDA。每当启动 GDA 进程时，它就创建一个新项，或者更新根目录的 **CDS_GDAPointers** 属性中的现有项。该项包含当前运行 GDA 的主机的地址。如果一个单元中有多个 GDA，每个 GDA 都创建并维护它们自己的 **CDS_GDAPointers** 属性中的地址信息。

当 CDS 服务器接收到不在本地单元中的名称的请求时，服务器会检查单元根目录的 **CDS_GDAPointers** 属性来查找一个或多个 GDA 的位置。第198页的图25 显示了 CDS 职员程序如何与 CDS 服务器交互来查找 GDA。

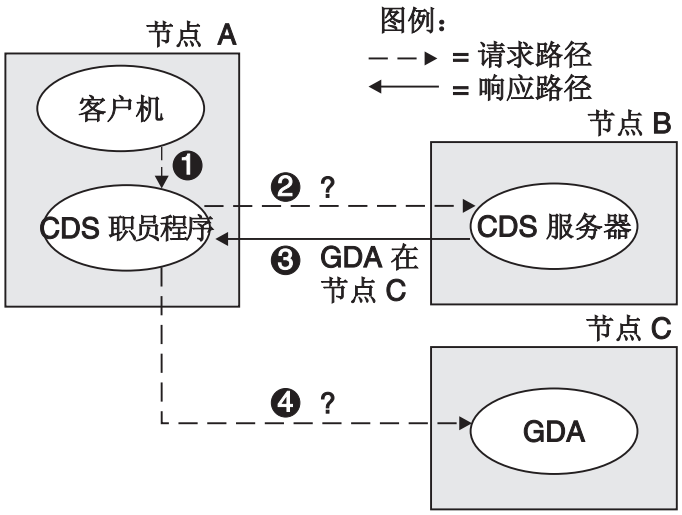


图 25. CDS 职员程序如何查找 GDA

以下步骤概述了上图中说明的 GDA 搜索:

1. 在节点 A 上，客户机应用程序将以 **/...** 前缀开头的全局名称传递给 CDS 职员程序。
2. 职员程序将查找请求传递到它知道的节点 B 上的 CDS 服务器。
3. 服务器的中央交换库包含了单元根目录的副本，所以服务器读取 **CDS_GDAPointers** 属性，并返回运行 GDA 的节点 C 的地址。
4. 职员程序将查找请求传递给 GDA。

图26 显示了 CDS 如何与 GDA 交互来查找 DNS 中定义的外部单元中的名称。假设名称是 **/.../widget.com/printsrv1**，它表示外部单元中的打印服务器。

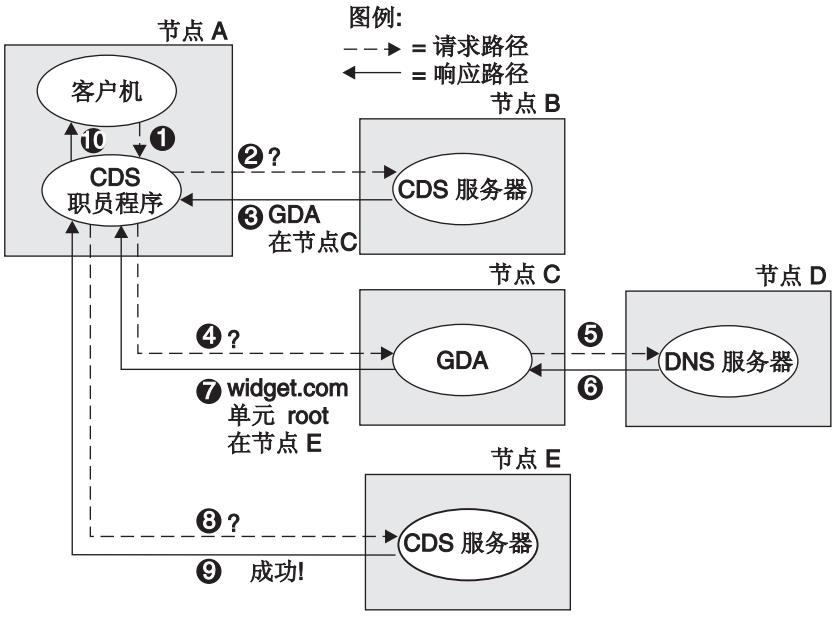


图 26. GDA 如何帮助 CDS 查找名称

以下步骤概述了上图中说明的名称搜索:

1. 客户机应用程序将名称 **/.../widget.com/printsrv1** 传递给 CDS 职员程序。

2. 职员程序将查找请求传递到它知道的节点 B 上的 CDS 服务器。
3. 服务器的中央交换库包含了单元根目录的副本，所以服务器查找 **CDS_GDAPointers** 属性，并返回运行 GDA 的节点 C 的地址。
4. 职员程序将查找请求传递给 GDA。
5. GDA 认出该名称是 DNS 样式的名称，所以它假设第二个组件是在 DNS 中定义的单元名称。它将名称 (**widget.com**) 的一部分传递到 DNS。处于简易的目的，该图只显示了一个 DNS 服务器；实际上可以包括多个 DNS 服务器来解析全局单元名称。

   注: 虽然这个示例涉及查找 DNS 样式的名称，操作的顺序和执行却与 X.500 名称或分层单元名称的操作几乎一致。如果 GDA 认出名称是 X.500 样式的名称，它会将该名称传递到 LDAP 客户机（经由 LDAP API）或 GDS 客户机（经由 XDS/XOM API），而不会传递到 DNS 服务器。然后，LDAP 客户机或 GDS 客户机与适当的服务器通信以获取单元绑定（与从 DNS 服务器获取的信息相同）。
6. DNS 进行查找，并返回与 **widget.com** 单元项关联的 GDA 信息。信息包括维护 **/.../widget.com** 单元名称空间根目录副本的服务器的地址。
7. GDA 将关于外部单元的信息传递给职员程序。
8. 职员程序与外部单元中节点 E 上的 CDS 服务器联系，将查找请求传递给它。
9. 节点 E 服务器的中央交换库包含了根目录的副本，所以服务器查找 root 中的 **printsrv1** 的项，并将请求的信息传递给节点 A 上的职员程序。为简便起见，该示例只显示与外部单元中的一个服务器联系的职员程序。解析全名时，职员程序实际上接收到对外部单元中几个服务器的引用。
10. 职员程序将信息传递给请求信息的客户机应用程序。

请注意，上两个示例（第198页的图25 和第198页的图26）表示初始查找。一旦 CDS 职员发现了 GDA 的位置，它就高速缓存此信息。职员程序还高速缓存它知道的外部单元中服务器的地址，以后请求相同单元中的名称时，启用它来直接联系外部服务器。

还要注意，GDA 知道它自己的单元名称，因此就可以避免联系全局目录服务来查找它自己单元中的名称。而且，GDA 可以识别单元名称是否符合 X.500 或 DNS 命名语法，并且它使用该知识将查找请求传递到适当的全局目录服务。 如果单元名称符合 X.500 命名语法，GDA 首先将请求发送到 LDAP 客户机，如果 LDAP 客户机／服务器不能解析它，然后再发送到 GDS 客户机。

管理全局目录代理

使用 DCE 配置程序来配置 GDA；一旦配置了 GDA 之后，它很少需要管理。（有关配置 GDA 的详细信息，请参阅 *IBM DCE 版本 3.2 AIX 和 Solaris 版: 管理指南--简介*。）

GDA 通常由作为普通系统启动和关机过程一部分而执行的脚本自动启动和停止。但有时可能要使用命令来停止并重新启动 GDA。用 DCE 配置程序配置了 GDA 之后，就可以使用这些步骤来启动和停止 GDA。

GDA 作为 **gdad** 进程运行。要启动 **gdad** 进程，使用以下步骤:

1. 确保 CDS 服务器已经在单元中运行。

2. 作为超级用户（**root**）登录系统。
3. 输入以下命令来启动 GDA 和所有必需的守护程序:

```
# start.dce gda
```

要停止 GDA，使用 **stop.dce gda**。

使其它单元可以查找您的单元

GDA 是允许本地单元中的 CDS 职员程序查找其它单元的一种机制。要使其它单元可以访问您的单元，必须在当前支持的全局命名环境中为单元创建一个项。这样做之前，从适当的命名管理机构中获取唯一的单元名称。（有关详细信息，请参阅 *IBM DCE 版本 3.2 AIX 和 Solaris 版: 管理指南--简介*。）

在配置单元、命名单元并初始化单元名称空间后，可以使用 **dcecp directory show** 命令来获取在 X.500、LDAP 或 DNS 中创建或修改单元项所需的数据。以下部分描述如何在 GDS、LDAP 服务器或 DNS 服务器中定义和维护单元项。这些部分假设您已基本熟悉 X.500 和 DNS；有关详细信息，请参阅每个全局名称服务的适当文档。

在域名系统中定义单元

DNS 中的名称与一个或多个叫作*资源记录*的数据结构相关联。资源记录存储在数据文件中，该文件的名称和位置都特定于实现。要创建单元项，必须编辑数据文件，并为每个维护单元名称空间 root 的副本的服务器创建两个资源记录。

第一个资源记录的类型可以在 AFSDB 或 MX，它包含了 CDS 服务器驻留的系统的主机名。可以使用 MX 作为使用 AFSDB 的备用方法。第二个记录的类型是 TXT，它包含了关于服务器维护的根目录副本的以下信息:

- 单元名称空间的 UUID，用十六进制记数法
- 副本的类型（主要或只读）
- 副本驻留的中央交换库全局 CDS 名称
- 中央交换库的 UUID，用十六进制记数法
- 中央交换库驻留的主机的 DNS 名称

以下示例显示了单元 **cs.tech.edu** 的一组 AFSDB 资源记录，其中有根目录的两个副本。请注意，只有第一个资源记录包含单元名称；第二、第三、第四个记录都假设为与同一个单元相关，因为它们都不包含单元名称。**TTL** 标题代表使用期限，它是一个值，以秒为单位，在它之后的数据都不再被认为在 DNS 高速缓存中是有效的。（所显示的值指定了缺省值 1 周。）**IN** 类表示协议是“因特网”，子类型 **2** 表示名称服务器在记录中命名的主机上。

```
;First Replica:
;Name            TTL       Class     Type     Subtype   Host
cs.tech.edu      604800    IN        AFSDB     2        fox.cs.tech.edu
;Name            TTL       Class     Type     Rdata
                 604800    IN        TXT      (1        ;version
                 fd3328c4-2a4b-11ca-af85-09002b1c89bb   ;ns uuid
                 Master                                 ;Replica1 type
                 /.../cs.tech.edu/cs1_ch                ;ch name
                 fd3328c5-2a4b-11ca-af85-09002b1c89bb   ;ch uuid
                 fox.cs.tech.edu)                       ;host
;Second Replica:
                 604800    IN        AFSDB     2        rox.cs.tech.edu
                 604800    IN        TXT      (1        ;version
```

```
                fd3328c4-2a4b-11ca-af85-09002b1c89bb   ;ns uuid
                Read-only                              ;Replica2 type
                /.../cs.tech.edu/cs2_ch                ;ch name
                fd3429c4-2a4b-11ca-af87-09002b1c89bb   ;ch uuid
                rox.cs.tech.edu)
;host
```

可以使用 MX 作为使用 AFSDB 的备用方法。以下示例显示了同一个单元 **cs.tech.edu** 的一组 MX 资源记录，其中有根目录的两个副本。

```
;First Replica:
;Name            TTL       Class     Type    Preference   Exchange
cs.tech.edu.     604800    IN        MX      1            fox.cs.tech.edu.
;Name            TTL       Class     Type    Rdata
                 604800    IN        TXT     (1           ;version
                 fd3328c4-2a4b-11ca-af85-09002b1c89bb     ;ns uuid
                 Master                                   ;Replica1 type
                 /.../cs.tech.edu/cs1_ch                  ;ch name
                 fd3328c5-2a4b-11ca-af85-09002b1c89bb     ;ch uuid
                 fox.cs.tech.edu)                         ;host
;Second Replica:
                 604800    IN        MX      2
rox.cs.tech.edu.
                 604800    IN        TXT     (1           ;version
                 fd3328c4-2a4b-11ca-af85-09002b1c89bb     ;ns uuid
                 Read-only                                ;Replica2 type
                 /.../cs.tech.edu/cs2_ch                  ;ch name
                 fd3429c4-2a4b-11ca-af87-09002b1c89bb     ;ch uuid
                 rox.cs.tech.edu)
;host
```

配置了单元之后，可以使用 **dcecp directory show** 命令来显示构造如上例中显示的那些资源记录所需的信息。以下示例 **directory show** 命令显示单元 **/.../cs.tech.edu** 的输出。

```
dcecp> directory show /.../cs.tech.edu
```

要在 DNS 名称空间中创建新的资源记录，使用来自 **directory show** 命令中的信息，并将正确格式化的数据放到 DNS 数据文件中。

在 LDAP 服务器中定义单元

ldap_addcell 实用程序获取 DCE 单元信息，并将此信息动态添加到 LDAP 服务器。必须用 root 权限来运行 **ldap_addcell** 命令。**ldap_addcell** 命令可以:

- 创建带单元绑定的新目录。
- 修改现有目录对象以添加单元绑定。
- 更改现有目录对象中的单元绑定的值。
- 删除现有目录对象中的单元绑定。

从目录对象添加或检索的单元绑定的格式与 X.500 服务器 (GDS) 的格式相同，并且存储在两个属性中:

- CDSCELL
- CDSREPLICAS

可以使用 **ldap_addcell** 命令将适当的单元信息添加到 LDAP 服务器。查找单元名称之后，GDA 将单元项中的数据传递到 CDS。然后，CDS 使用该信息来联系单元中的服务器。

认证信息，如 userid 和 password，都是 ldap_addcell 实用程序调用的一部分，因为它将写到目录服务。无论是用 X.500 注册实用程序还是用 ldap_addcell 注册实用程序写，存储在目录服务中的 DCE 单元信息都相同。

ldap_addcell 命令有以下语法:

```
ldap_addcell -h ldap_server -a authentication_DN -p password [-o object_class, \
object_class...]|[-d]
```

其中:

-h *ldap_server*
用于保存单元绑定的 LDAP 服务器的名称。

-a *authentication_DN*
将认证的以 LDAP 名称语法指定的专有名称 (DN)，并用于添加单元绑定。

-p *password*
用于认证专有名称 (DN) 的口令。

-o *object_class*
要创建或修改的项（注册）的属性 object_class 的值。请注意，如果列出了多个 *object_class* 值，必须用逗号将它们隔开。

-d 从目录中的项中删除 DCE 单元信息属性。它并不除去整个目录项。

必须用 root 权限来运行命令，而且命令将消息打印到 stderr。

以下的 **ldap_addcell** 示例假设:

- bermuda.austin.ibm.com 是 LDAP 服务器设备名称。
- gdatest 是对 LDAP 服务器有写访问权的用户。
- gdatest 还是用户 gdatest 的口令。
- 允许 organizationalUnit 包含辅助对象 dceCellInfo。
- LDAP 服务器执行模式检查。

示例 1 显示了在 LDAP 服务器中正常创建单元绑定。

```
ldap_addcell -h bermuda.austin.ibm.com -a "cn=gdatest,ou=austin,o=ibm,c=us" -p "gdatest" \
       -o organizationalUnit,dceCellInfo
```

示例 2 显示了删除 CDS_CELL 和 CDS_REPLICAS 属性。

```
ldap_addcell -h bermuda.austin.ibm.com -a "cn=gdatest,ou=austin,o=ibm,c=us" -p "gdatest" -d
```

示例 3 显示了更改已存在的对象中的 CDS_CELL 和 CDS_REPLICAS 属性。

```
ldap_addcell -h bermuda.austin.ibm.com -a "cn=gdatest,ou=austin,o=ibm,c=us" -p "gdatest
```

ldap_addcell 命令的大多数参数都有相应的环境变量，如果在 **ldap_addcell** 命令调用中没有出现参数，那么将使用相应的环境变量。**ldap_addcell** 参数及其相应环境变量是:

ldap_addcell 参数	**环境变量**
-h	LDAP_SERVER
-a	LDAP_AUTH_DN
-p	LDAP_AUTH_DN_PW
-o	LDAP_OBJECT_CLASS

注: -d 参数没有相应的环境变量。

如果已经注册了单元项，那么新的值将替换此单元的 CDS_CELL 和 CDS_REPLICAS 属性，除非指定了 **-d** 参数。

由 CDSCELL 和 CDSREPLICAS 替换的 CDS_CELL 和 CDS_REPLICAS

由于 LDAP 协议版本 3 不支持下划线 (_) 作为 CDS_CELL 和 CDS_REPLICAS 的合法字符，所以它们被分别更改成 CDSCELL 和 CDSREPLICAS。如果已经使用 DCE 2.2 的初始版本将 X.500 单元名称注册到 LDAP 服务器中，则必须在每个运行 **gdad** 服务器的系统上安装 DCE 2.2 最新版本。然后使用 **ldap_addcell** 命令来重新注册 DCE 单元信息。**ldap_addcell** 必须在已经升级到 DCE 2.2 的最新版本的系统上运行。LDAP 服务器中的模式也必须更新成使用 CDSCELL 和 CDSREPLICAS 来分别代替 CDS_CELL 和 CDS_REPLICAS。

GDA 不支持的 SLAPD 引证

独立 LDAP 守护程序 (SLAPD) 支持的引证当前有以下限制，这些限制可以防止 GDA 使用它们来解析外部单元绑定:

由 **ldap_search_s( )** API 使用的 SLAPD 引证要求搜索作用域是 LDAP_SCOPE_SUBTREE。换句话说，必须搜索引用点下的整个子树。GDA 知道它需要的特定对象，并在作用域 LDAP_SCOPE_BASE 中进行搜索。GDA 不能从 **ldap_search_s( )** 的结果中接收多个对象。

第5部分 DCE 分布式时间服务

第23章 DCE 分布式时间服务介绍

DTS 是基于软件的服务，它为局域网 (LAN) 和广域网 (WAN) 中的系统提供精确且容错的时钟同步。DTS 提供的时钟同步使分布式计算应用程序可以确定事件的先后顺序、持续时间和时间安排。

DTS 由合作系统组的软件组件组成；它符合 DCE 中使用的客户机／服务器模型。在 DTS 实现中，每个服务器都通过中间件*职员程序*来向许多客户机系统和应用程序提供时间。职员程序驻留在客户机系统上。（请注意，在本指南的这一部分中，术语*实体*是指拥有相同功能的服务器进程或职员进程。）

大多数 DCE 节点都有 DTS 职员程序，用于调整客户机系统上的时钟；职员程序使用远程过程调用 (RPC) 从网络中的一个或几个服务器处获取时间值。没有 DTS 职员程序的节点都拥有 DTS 服务器；除了向职员程序提供时间值以外，服务器还调整其主机系统上的系统时钟。服务器还可以从网络外部的标准时间源获取参考时间值。

因为没有设备可以随时测量确切时间，DTS 用一个包含正确时间的间隔来表示时间。在 DTS 模型中，职员程序从几个服务器中获取时间间隔，并计算这些间隔重叠的交集。然后，职员程序将其客户机系统的系统时钟调整到计算出的交集的中点。如果职员程序接收到一个时间间隔与多数时间间隔不相交，那么职员程序就宣称该不相交的值是错误的。职员程序在计算新的时间时忽略错误的值，因此就确保了错误的服务器时钟不会影响到客户机。

DTS 还允许从外部来源导入时间值，如美国国家标准技术学会 (NIST)。 DTS 使用 UTC（协调世界时间）标准，这种标准已经在很大程度上代替了格林威治标准时间 (GMT) 而成为参考标准。许多标准主体通过广播电台、电话和卫星传播 UTC；商业设备（时间提供者）可用于接收和解释这些信号。DTS 提供了一个时间提供者接口 (TPI)，它描述了时间提供者进程如何将 UTC 时间值传递到 DTS 服务器，并在网络中传播它们。TPI 还允许其它分布式时间服务与 DTS 同时操作。

DTS 为运行分布式应用程序的计算机网络提供了其它有价值的服务。DTS 的主要特性和优点如下:

- 正确性--DTS 使客户机接收正确时间的可能最大化。DTS 使用 UTC 作为基本参考，并定义了任何包含正确 UTC 的时间间隔。
- 定量时间测量--DTS 使用特定测量和制造商的规范来确定服务器报告的时间的质量。
- 容错--DTS 报告错误的服务器，并且在时钟同步期间不使用它们的时间值。
- 管理能力--DCE 控制程序 (**dcecp**) 使您可以控制和监控软件。
- 应用程序编程接口 (API)--DTS 提供了一个接口，允许应用程序获取、比较和计算 UTC 时间值。
- 本地时间转换--显示时间值时，DTS 将它内部使用的 UTC 时间转换成本地时间值。
- 单一性--DTS 通常提供单调的时钟调整。可以使用 DCE 控制程序进行单调的时钟调整。
- 自动配置--DTS 实体使用 RPC 概要（搜索表）来获取本地区域或单元中服务器的位置。
- 效率--复杂性在服务器中考虑；网络开销很小。

DTS 优点

DTS 提供通常时间服务所提供的所有特性，它还提供了几种增强网络性能的特性。以下部分描述了这些 DTS 特性:

- 应用程序支持
- 外部时间提供者支持
- 可管理性
- 定量精度测量

应用程序支持

操作系统和分布式应用程序需要同步时间测量来协调它们的进程。在网络中，DTS 相互之间同步系统时钟，如果使用外部时间提供者，则与 UTC 时间标准同步。任何读取系统时钟的分布式应用程序都需要 DTS，而大多数应用程序皆属此类。随着网络中的分布式应用程序和系统日益增加，DTS 对于进程协调也日趋重要。

DTS 向系统时钟提供了几种现有的使用同步时间的应用程序。这些应用程序必须参考同步的系统时钟来协调在网络中发生的事件。应用程序使用同步的时钟来执行以下功能:

- 事件测量--应用程序可以读取系统时钟来启动和停止定时器，并测量事件之间的时间间隔。
- 事件报告--应用程序可以在事件发生时读取时钟，并将时间戳记附加到事件报告中。
- 事件安排--应用程序可以读取系统时钟并添加相对时间以确定将来事件的发生。
- 事件顺序--应用程序可以通过读取同步的系统时钟派生的时间戳记来确定事件的顺序。

对于新的应用程序，DTS 会提供一个 API。该 API 提供了新应用程序可以用于获取和操作二进制时间戳记的例程。DTS API 支持 ANSI C 语言结构。（有关 DTS API 的进一步信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Application Development Guide--Core Components*。）

外部时间提供者支持

大多数网络都希望系统时钟与 UTC 时间标准保持同步。许多商业设备都可以获取由标准组织提供的 UTC 时间；这些设备通过短波无线电、卫星和电话接收信号。如果网络或单元包含不止一个 LAN，那么建议 DTS 软件使用至少一个外部时间提供者。（有关时间提供者硬件的供应商列表，请参阅附录 C。*dcelocal*/**usr/examples/dts** 中提供了在线样本时间提供者程序。）

DTS 服务器能够以 TPI 方式（*IBM DCE Version 3.2 for AIX and Solaris: Application Development Guide--Core Components* 中有描述）与时间提供者保持同步。TPI 指定了 DTS 服务器进程和时间提供者进程之间的通信。

当 DTS 服务器试图同步时，它使用 TPI 来检查时间提供者进程。如果有可用的时间提供者，则服务器只与该时间提供者同步。如果没有时间提供者，则服务器与网络中的其它服务器同步。

通过将时间提供者与 DTS 服务器结合使用，可以确保服务器与 UTC 密切同步。当其它服务器从带时间提供者的服务器（TP 服务器）请求时间时，TP 服务器的精确时间将传播到整个网络。（有关时间提供者和服务器同步进程的进一步信息，请参阅『基本 DTS 概念』。）

可管理性

DTS 同步功能作为后台处理运行；一旦配置了 DTS 之后，系统管理员同步系统时钟几乎或根本不需要输入。DTS 还是可容错的。它能防止出故障的时钟向网络中的其它时钟提供错误时间。然而，系统管理员偶尔可能需要执行以下功能:

- 标识系统时钟问题
- 调整系统时钟
- 根据不同的网络条件更改 DTS 属性
- 网络拓扑更改时修改系统配置

DTS 提供了功能完整的管理界面，该界面允许系统管理员调整系统时钟、更改 DTS 管理参数的值，以及将服务器添加到网络或从网络中除去服务器。

为解决系统时钟的问题，DTS 提供了事件报告，它可以通知系统操作员和管理员诸如系统时钟不准确或无法同步等罕见事件。

定量精度测量

与其它网络时间服务不同，DTS 使用制造商的规范和直接观察来确定系统时钟相对于 UTC 的不准确性。DTS 将精度测量附加到它内部使用的每个时间值。该测量要考虑累积的时钟错误、通信延迟和处理延迟。DTS 使用一个或几个来源的时间和精度测量组合为客户机系统计算最准确的新时钟设置。（有关 DTS 同步进程的进一步信息，请参阅第211页的『同步化系统时钟』。）

基本 DTS 概念

以下部分描述了系统时钟和网络特征、DTS 同步概念、DTS 时钟调整和 DTS 时间表示法。系统管理员在阅读第225页的『第25章 管理 DCE DTS』之前，需要阅读这些部分以初步了解 DTS 概念。

时间测量因素

以下几部分描述了影响时间测量的因素，并解释了 DTS 如何处理它们。

时钟错误

所有系统时钟都有导致时钟错误并干扰同步进程的公共特性。系统时钟错误往往会增加超时；错误的更改速率就叫作*漂移*。如果网络中的每个系统时钟都在同一时刻开始，并且按相同速率运行，那么这些时钟就保持同步。然而，因为每个系统时钟都按不同的速率漂移，整个网络中的系统时钟就变得不同步。

任何两个时钟之间的差异就是这两个时钟之间的*偏差*。许多计算机系统中使用的时钟都有一个特定漂移，最大为每天几秒。如果几天没有校正，连网的系统时钟之间的偏差就可能影响分布式应用程序的性能。

每个节点上的 DTS 服务器或职员程序都会跟踪其客户机系统时钟的漂移，并定期与其它 DTS 节点同步，以减少其自身与其它 DTS 节点的客户机时间值之间的偏差。在这个重复的同步进程中，DTS 服务器或职员程序调整其客户机节点上的系统时钟是最后一步。

通信和处理不确定性

通信延迟也约束了同步化进程，特别是当两个通过 WAN 或低速链接进行通信时。DTS 可以调整在系统间发送和接收消息所必需的已知处理延迟。但是，由于通信链路的质量不同，所以对于不同消息，用于发送、接收和确认消息的时间也各不相同。不能精确地知道这些延迟，因为网络传输和要求读取进入时间戳记的时间都在不停变化。

DTS 记录下与时间测量一起在网络上发送的所有已知错误因素，而不是使用通信和处理延迟的估计值。该测量使 DTS 可以确定时间源的相对质量，而不考虑它的地理位置或通信链路上变化的条件。

不准确性值

为了使系统时钟与最准确的设置保持同步，DTS 需要一种方法可以确定相对于彼此之间或 UTC 的时间源的准确性。本节描述 DTS 如何确定网络上任何可用时间源的相对准确性。

DTS 使用不准确性值或误差来确定它从系统时钟和外部时间提供者获取的时间值的精度。这个 DTS 特性将每个时间值有效变换成间隔或范围，而不是区间中的一个点。

通过以下三个因素确定不准确性值:

- 漂移 --读取时钟时，DTS 计算自上次读取时钟后时钟漂移的时间量。漂移是大多数不准确性值的最大组成部分。
- 通信延迟--该不准确性还包含系统间通信延迟的不确定部分。虽然 DTS 补偿处理延迟，但它不能预测或直接测量网络链接上发生的不断变化的延迟。职员程序或服务器从 LAN 上协同定位系统中获取的不准确性值要比从 LAN 之外的服务器中获取的不准确性值小得多。
- 跳跃秒数 --UTC 时间是由原子钟测量的，它极其稳定。然而，标准使得时间还是基于地球的位置。由于地球转动不断变慢，有时需要将 UTC 提前 1 秒。这些事件就是*跳跃秒数*。跳跃秒数可能发生在任何一个月的最后一秒，通常每 18 个月发生一次。在每个月的月尾，DTS 通过对所有精度测量增加 1 秒来解决跳跃秒数问题。如果外部时间提供者确定跳跃秒数其实并未发生，DTS 稍后会调整时钟以除去额外的不准确性秒数。

如果 DTS 不做校正，那么系统时钟的不准确性就会不断增加。例如，假设时钟在 UTC 时间 0:00:00.00（午夜）启动，并且不准确性为 0。由于漂移，当时钟下一次显示 0:00:00.00 时，不准确性为 8 秒。UTC 时间也许是 23:59:52.00 或 0:00:08.00，但也可能是两者之间的一个值。因此，系统时间是一个包含 UTC 且由不准确性为界的间隔，如 第211页的图27 所示。如果使用 DTS 格式来显示时间，那么组合了时间和不准确性间隔显示如下:

```
1993-08-03-00:00:00.000I08.000.
```

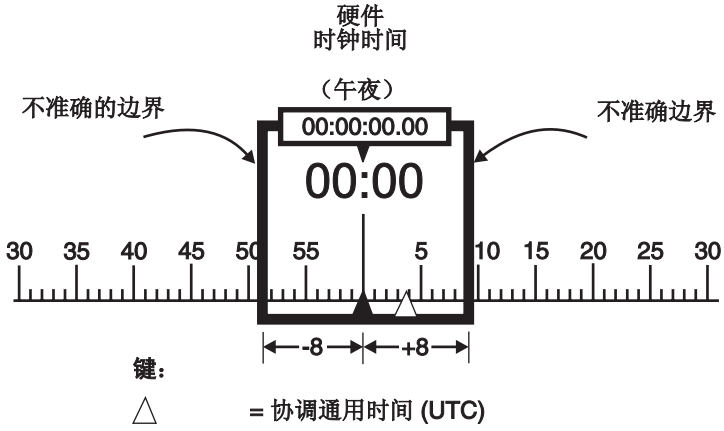


图 27. 时间和不准确性

同步化系统时钟

要维护一致的系统时间，DTS 服务器和职员程序需定期同步化所有网络系统中的时钟。每个系统上的 DTS 实体通过请求服务器将组合的时钟和不准确性值（时间间隔）发送给原来的系统来执行这些同步。然后实体使用服务器发送的值来计算新的系统时间。

DTS 服务器和职员程序的同步过程稍有不同。在试图与其它系统同步之前，DTS 服务器总会检查服务器系统上是否有外部时间提供者。如果没有时间提供者，一个特定服务器会从其它服务器请求时间。如果没有可用的时间提供者，且服务器与其同级服务器同步，当服务器计算新的系统时间时，它会使用自己的系统时间作为一个输入值。

大多数网络系统都运行 DTS 职员进程。职员程序不能有时间提供者，它们不使用客户机系统的系统时间来计算新的时间。当职员程序与其客户机系统的时钟同步时，职员程序只使用从服务器获取的时间值来计算新的系统时钟。

当 DTS 职员程序从几个服务器请求时间间隔时，它使用它们来计算新的时间，该时间是正确的（即，包含 UTC），且将不准确性降到最低。服务器响应后，且 DTS 职员程序计算了网络通信不确定性和每个时间值的漂移之后，职员程序就有了一组间隔（第212页的图28 中的 **t1** 到 **t4**）。由于每个间隔都包含 UTC，所以交集就是职员程序可以选择的包含 UTC 的最小间隔。这个交集就是计算时间。DTS 实体使用计算时间间隔来调整接收服务器值的系统上的时钟。

除了在同步期间消除大的不准确性值，DTS 还弃用从错误时钟接收到的间隔（图中的 **t2**）。DTS 检测并拒绝与多数间隔不相交的时钟间隔。但 DTS 检测到错误间隔时，它通过显示一条事件消息来通知系统管理员，标识出发送了错误值的服务器。

具有高漂移时钟或网络中远处的服务器将它的时间提交给 DTS 实体（图中的 **t1**），但由于有更准确的时间，将忽略较大的时间间隔。 请注意，在第212页的图28 中，正确时间 (**t1**) 的端点到计算时间中点的距离比宣称为错误的间隔 (**t2**) 端点到计算时间中点的距离更远。

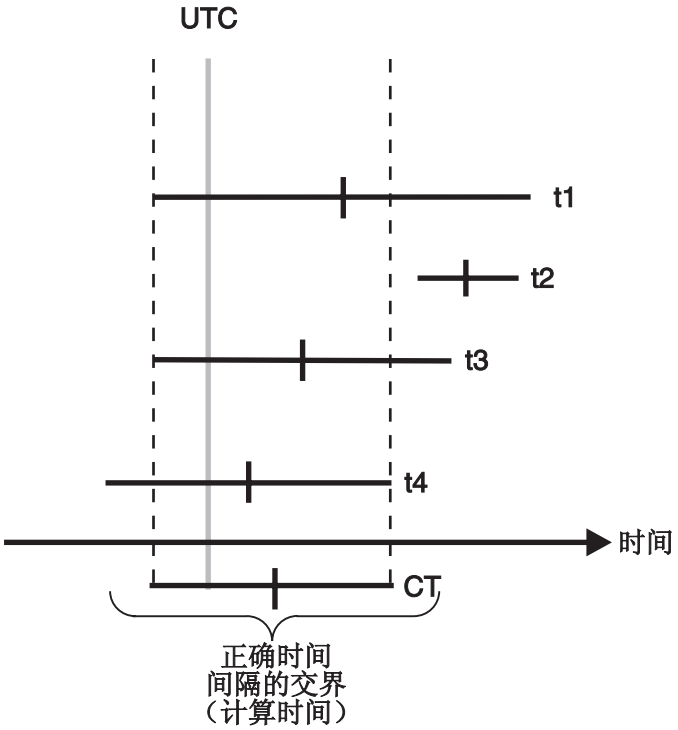


图 28. 计算时间

同步进程中，拥有最大准确性的服务器对确定整个网络的新系统时间的影响也最大。上一个图中，提交时间值 **t3** 的服务器拥有最小的校正间隔，因此最接近计算时间。带外部时间提供者的服务器系统通常是具有最精确时间的服务器。除 TP 服务器之外，拥有最高质量时钟和最好通信链路的服务器会最大程度地影响其它系统上的时间。

同步进程还减少了系统间的偏差。计算时间间隔通常小于任何一个时钟提供的间隔。请注意，上图中的计算时间就小于任何一个源间隔。由于同步过程在每个网络系统中不断重复，系统间的偏差也逐渐减小，因此它们越来越接近同步。但是，如果网络中没有时间提供者，时钟也许都会偏离 UTC。

DTS 如何调整系统时钟

许多系统时钟都基于振荡器，并且结合硬件和软件一起运行。每个时钟的硬件都包含定时器，它按固定间隔向操作系统发送中断信号；每个中断信号就是一个滴答。每次滴答时，都会向包含当前时间值的软件注册器增加一个固定量（例如，10 毫秒）。DTS 只需更改添加到软件注册器的增量值就可以调整时钟的速率。它不会直接影响硬件时钟的滴答。

DTS 按 100 比 1 的比例来调整系统时钟；即，它需要 100 个时间单位来调整 1 个时间单位的错误。例如，它用 1 分 40 秒的时间来校正 1 秒的错误。这个调整速率超过了正常的漂移速率，因此时钟不会对同步产生严重的干扰。

第213页的图29 说明了 DTS 如何更改软件注册器的增量。最上面一行表示每隔 10 毫秒滴答都将正常时钟增加 10 毫秒。中间一行说明了对快速时钟的调整；DTS 在每次滴答时将注册器增加 9.9 毫秒，而不是 10 毫秒，这样来调慢时钟。最下面一行说明了对

慢速时钟的调整；DTS 在每次滴答时将注册器增加 10.1 毫秒，而不是 10 毫秒，这样来调快时钟。

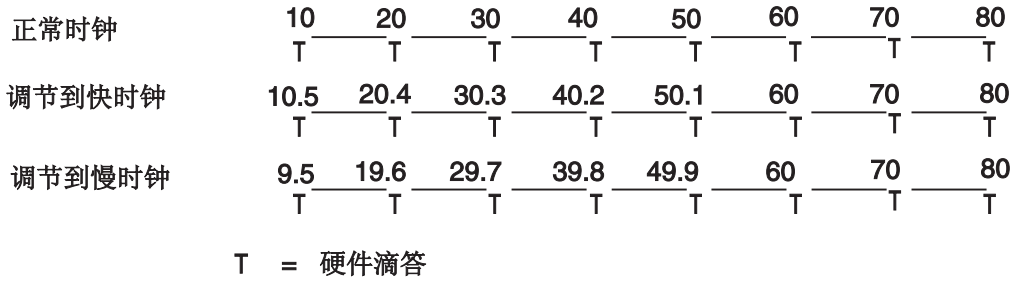


图 29. 时钟调整

有时应立即设置系统时钟，而不是逐步调整它。在以下情况中，DTS 提供这个选项:

- 如果在系统启动期间想要设置初始系统时间
- 如果上一次同步在很久以前，并且判定系统时钟之间的偏差已经太大，等不及逐步调整
- 如果网络有灾难性的硬件问题，导致许多时钟都产生错误时间
- 如果给定时钟的时间间隔与其它时钟的间隔不相交，而且错误超出了预定的允许程度

DTS 时间表示法

UTC 是国际时间标准，它很大程度上代替了 GMT。该标准由国际时间局 (BIH) 管理，并已得到广泛使用。对于其所有内部进程，DTS 使用表示 UTC 的不透明二进制时间戳记。不能读取或分解 DTS 二进制时间戳记。DTS API 允许其它应用程序转换或操作时间戳记，但却不能显示。DTS 还将时间戳记转换成 ASCII 文本，以便在客户机系统上显示。

绝对时间

绝对时间是时间刻度上的一点。对于 DTS，绝对时间参考的是 UTC 时间刻度。绝对时间测量从系统时钟或外部时间提供者处派生而得。当 DTS 读取系统时钟时间时，时间记录到不透明的二进制时间戳记中，该时间戳记还包括不准确性和其它信息。当使用 DCE 控制控件 (**dcecp**) **clock show** 命令来显示绝对时间时，该时间转换成 ASCII 文本，显示如下:

```
1993-11-21-13:30:25.78523-04:00I010.0825
```

DTS 以符合 ISO 的格式显示所有时间。第214页的图30 中详细显示了上一个显示示例中生成的国际标准组织 (ISO) 格式。

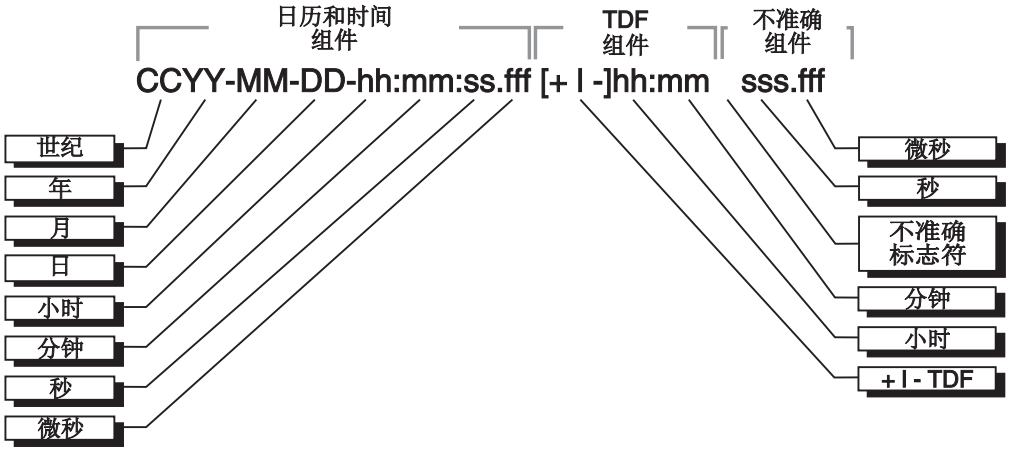


图 30. 符合 ISO 的时间格式

在上图中显示的格式示例中，由 **+**（加号）或 **-**（减号）开头的相对时间表示日历日期和不准确性偏离 UTC 的小时和分钟数。在字符串中出现这两个字符之一还表示日历日期和时间都是系统的本地时间，而不是 UTC。描述符 I 表示与时间关联的不准确组件的开头。可以用几种方法来表达要显示的 DTS 时间。第469页的『附录D. DTS 扩展的 BNF』 中定义了 BNF 格式的 DTS 时间。

虽然 **dcecp clock show** 命令用以上格式显示了所有时间（请参阅图30），接口还接受以下 ISO 格式输入的变体，如图31 显示。

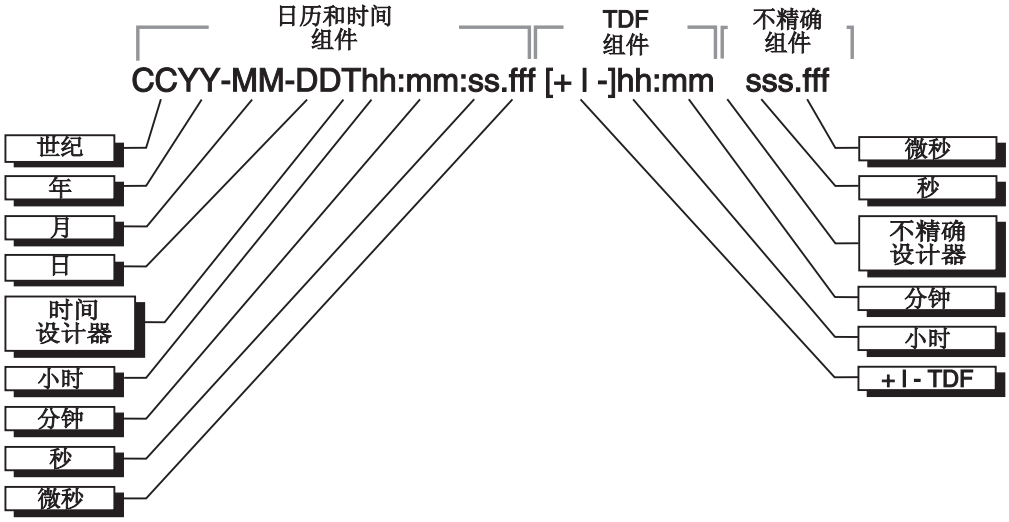


图 31. 符合 ISO 的时间格式变体

在上例中，描述符 T 将日历日期与时间隔开，″**,**″（逗号）将秒与秒的小数部分隔开，±（加号或减号）表示不准确组件的开头。

DTS 提供了一种转换功能，每当显示时间时，它会将基于 UTC 的绝对时间更改成本地时间。所显示的本地时间由 UTC 将时间差别因子 (TDF) 派生而来，TDF 可能有正负号。在上例中，字符串 [**+/-**] *hh*:*mm* 表示 TDF。安装系统时，为系统选择时区规则，该规则确定了 TDF 以及 TDF 的季节性更改。初始启动后，所有随后的输出时间都反映了本地时间。如果系统显示绝对时间，并且不包含 TDF 信息，那么这就是 UTC 时间。

相对时间

相对时间是一个离散的时间间隔，它通常用于加上另一个时间，或从另一个时间减去。与绝对时间关联的 TDF 就是相对时间的一个例子。相对时间通常用作命令或系统例程的输入。

图32 显示了相对时间的格式。

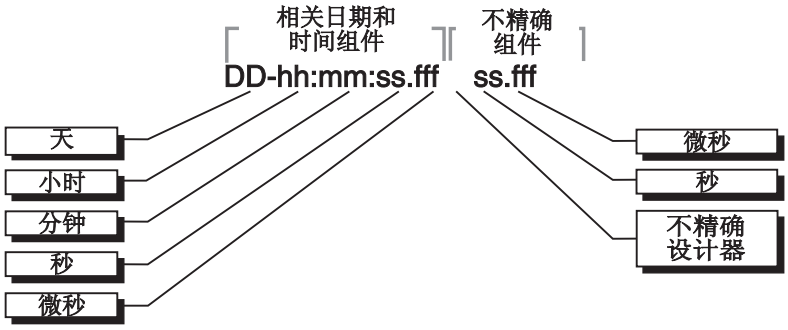


图 32. 相对时间格式

用 DTS 相关的 **dcecp** 命令指定的简单相对时间既不使用日历日期，也不使用不准确性字段，因为这些字段与绝对时间相关。正的相对时间不带符号，但负的相对时间前面加一个 **-**（负号）。

下例显示了用典型的 DTS 相关的 **dcecp** 命令中使用的相对时间:

```
21-08:30:25.000
```

简单相对时间通常在其它相对或绝对时间的基础上做加减运算。例如，如果要说，『我要在一小时之内见到你』，那么就是在现在绝对时间上加上相对 +01:00。如果要添加或减去一个相对时间和绝对时间，请注意，输入绝对时间的不准确性会转移到生成的绝对时间。例如，1993-11-30-00:30:25.000I00.030 减去 00-00:15:25.000 等于 1993-11-30-00:15:00.000I00.030。

DTS 工作原理

DTS 有两个主要的软件组件:

- 职员程序
- 服务器

以下部分描述了所有这些组件，并告诉您它们如何交互来向客户机应用程序提供时间，以便同步系统时钟。

职员程序

任何系统不是 DTS 服务器就是 DTS 职员程序。大多数网络系统都运行职员程序软件。职员程序维护服务器列表，并对 DTS 客户机系统执行同步功能。

为了构建服务器列表，并且与列表中的服务器同步，职员程序应能够自动查找服务器。它们使用远程过程调用 (RPC) 概要来查找服务器。回忆一下，概要是包含以下类型的项的搜索表:

- 服务器项--个别资源提供者的 CDS 名称。

- 服务组项--由单一 CDS 名称标识的一组资源提提供者。
- 概要项--其它配置概要的名称。这些项允许概要的分层嵌套。

每个 DTS 职员程序节点包含最多三个概要。当职员程序试图查找服务器时，它首先对基本概要*节点初始概要*中的项执行 RPC 查找。然后职员程序查找 LAN 概要项。如果没有找到 LAN 概要项，职员程序将搜索缺省概要项；缺省概要项可能会包含 LAN 概要项。但职员程序查找 LAN 概要时，它读取服务器项来构建本地服务器的列表。这个进程将按已设置的时间间隔重复进行。

如果职员程序没有获得如 DTS 管理属性 **minservers** 规定的那样足够的服务器项，它将尝试查找附加服务器，通常是 LAN 以外的服务器。为查找这些服务器，职员程序查找单元概要，此概要有众所周知的 CDS 名称。单元概要包含全局服务器项；即，通常在 LAN 之外找到的服务器。（有关服务器的进一步信息，请参阅『服务器』。）

用足够的项构建了服务器列表之后，职员程序可以直接从列表上的几个服务器请求时间值。然后职员程序接收这些时间值，并使用它们来为客户机系统计算新的系统时间。

服务器

服务器为 DTS 提供了许多通信和同步功能。就像职员程序一样，它们导入 LAN 和单元概要中其它服务器的信息。但是，服务器还将绑定导出到它们自己的 CDS 名称空间项中，并将它们的名称导出到 LAN 和单元概要中。（有关如何配置和查找服务器的进一步信息，请参阅以下关于服务器子类型的部分。）

外部时间提供者可以连接到服务器，它们会将从时间提供者处获取的精确时间间隔传播到整个网络。

在一个服务器可以从另一个服务器中获取时间值之前，这两个服务器必须拥有相同的时段号。时段将 DTS 实现在逻辑上分成单独的区域。服务器只与其它拥有相同时段号的服务器同步。所有服务器在创建时拥有相同的时段号。偶尔，您可能希望使用管理界面来更改服务器的时段号，将它与网络隔离，以便改正问题。

本地服务器集合

本地服务器驻留在同一个 LAN 上，并通过互相同步来维护它们的时钟。由于此类网络的高吞吐量，LAN 中本地服务器之间的偏差通常维护在 200 毫秒之内。如果本地集合中至少有一个服务器与精确的时间提供者同步，那么每个服务器上的不准确性都会减少。

第一次初始化服务器时，它将绑定导出到它在名称空间中的项，并将它的名称项添加到 LAN 概要。每个服务器都会自动输入网络相关部分的 LAN 概要中。本地服务器还会从 LAN 概要中导入绑定以构建它们可以同步的服务器的列表。

本地服务器执行时间间隔计算、调整它们的时钟，并且互相提供时间值以便同步。每个服务器都试图定期与本地集合中的其它所有服务器同步。职员程序则按更长的间隔从本地服务器请求时间值。但是，职员程序只需要从一定数量（由 **minservers** 属性确定）的服务器请求间隔，这些服务器通常是所有本地服务器的一个子集。

全局服务器集合

本地服务器只对于一个 LAN 中的服务器和职员程序可用，但全局服务器在整个单元中都可用。任何服务器都可以配置成本地服务器或全局服务器（请参阅 DCE 控制程序 **dts configure** 命令）。全局服务器的数量提出很少，但全局服务器有一些重要的功能可以启用 DTS 同步网络中的每个节点。在以下情况中，全局服务器是必需的:

- 网络有多个 LAN 或者一个扩展 LAN
- 不在 LAN 上的系统通过点到点链路可以访问 LAN
- 如果职员程序或本地服务器不能访问一定数量的本地服务器，该数量由 **minservers** 属性确定

通过使用 **dcecp dts configure** 命令并选用 **-global** 选项可以将本地服务器重新配置成全局服务器。将服务器配置成全局服务器会使该服务器将其绑定导出到它在名称空间中的项，并将它的名称导出到单元概要中。

当本地服务器和职员程序无法获取一定数量（由 **minservers** 属性确定）的本地服务器响应时，它们将从全局服务器请求时间值。某些本地服务器还会定期从全局服务器请求时间。

信使

每次同步时，称作*信使*的本地服务器从任意选择的全局服务器请求时间值。当 DTS 启动时，它自动将服务器的 **courierrole** 属性值设置为 **backup**。可以通过手工更改这个属性值来更改服务器的信使角色。要这样做，使用 **dcecp dts modify** 命令并选用 **-change** 选项。如果服务器连接到外部时间提供者，要将它重新配置成信使。

信使维护全局服务器的列表，它们从单元概要中导入这些全局服务器的绑定。每次同步时，当同步它们自己的时钟时，信使使用所有本地服务器和一个全局服务器的响应。信使通过以下过程提供整个网络的同步:

1. 信使至少从远程区域中的一个全局服务器请求时间值，并从一些本地服务器中请求其余的值，本地服务器的数量由 **minservers** 属性确定。
2. 信使使用全局服务器时间和本地服务器时间来同步它们各自系统中的时钟。
3. 在将来同步期间，信使将最新计算的时钟时间传递给 LAN 上的其它服务器和职员程序。

对于包含多个 LAN 或点到点链路的网络，需要将每个 LAN 或每一段上的一个服务器配置成信使。这个配置确保了网络的各部分都保持同步，并且不会相互隔离。

通过使用管理界面，您还可以指定一个或多个服务器作为备份信使。在 LAN 上没有可用信使服务器的事件中，这些本地服务器临时充当信使。这种情况下，带有最低编号的全局唯一标识符 (UUID) 的备份信使定期与全局服务器同步，直到信使再次可用。

如果信使不能找到任何全局服务器，那么它将使用本地服务器，并增加它的**没有检测到全局服务器**计数值。

第24章 规划 DTS 实现

与 DTS 软件交互的人员主要有两类: 系统管理员和应用程序员。程序员一般不需要参与 DTS 实现的规划阶段。但是如果正在等待程序将一个 UTC 时间导入服务，则一定希望把时间提供者放在与程序员最接近的服务器上。与时间提供者的靠近对程序员使用时间供应者硬件来测试软件应用程序非常有帮助。

一般，系统管理员或网络工程师将规划 DTS 实现。他们将决定哪些节点是服务器那些是职员程序，并决定 DTS 实现如何随网络的扩展而增长。DTS 对大型网络而言是规模可伸缩的，所以要扩充该实现加入新的节点是十分容易的。

系统管理员还需安装软件以及维护 DTS。当网络扩展时，系统管理员应确保服务正以可接受的准确性运行并安装新的服务器、时间提供者和职员程序。

常规规划指南

在规划 DTS 实现时，请考虑以下问题:

- 您的单元是单一 LAN、扩展 LAN、WAN 还是 LAN 与 WAN 的组合？
- 当前或希望的网络拓扑结构（组件位置）是什么？
- 需要多少服务器？要放在哪里？
- 是否需要全局服务器？要放在哪里？
- 如果正在使用全局服务器，是否需要配置信使？
- 是否使用外部的时间提供者来获取 UTC？

以下部分将帮助您回答这些问题。

虽然有许多网络配置会影响 DTS 规划，但有些通用规则应用时无需考虑网络配置或网络中的节点数。以下将概述这些指南:

- DTS 必须和其它 DCE 组件安装在一起。
- 找到在任何可能的情况下，与其它 DCE 组件的服务器在同一节点上的 DTS 服务器。
- 每个单元至少应该有三个 DTS 服务器；最好是四个服务器以提供冗余。
- 每个 LAN 中应至少有一个服务器。
- 找到具有最多节点数的位置上的服务器。

虽然在规划网络时还必须考虑其它一些因素，但是这些因素取决于网络的拓扑结构和配置。以下部分将介绍一些有助于您在自己的网络中实现 DTS 典型的单元排列。

LAN 的 DTS 配置

如果节点是在单一 LAN 中，则无需考虑节点数，规划 DTS 实现将十分简单的。要检测出错时间服务器，至少要把三个系统配置成服务器。如果要为 DTS 实现提供冗余，则应规划在网络中安装四个或更多服务器。用这种方法，即使其中一个服务器失败，DTS 仍然可以可靠地同步。

要确保 DTS 实现的可靠性，应确认服务器节点之间的网络连接是稳定的。如果计划把 WAN 链接添加到 LAN 中，请不要把服务器移到远程节点，因为 WAN 链接通常没有 LAN 的可靠。

如果是在单一 LAN 中，服务器在 LAN 中的位置无关紧要。您可以把其中一个服务器放到十分容易访问到节点上以帮助进行故障检测，除此之外，不再建议其它服务器位置。既不需要全局服务器也不需要信使。

如果计划使用一个或多个时间提供者，请把它们放到最容易访问到的系统上，以方便启动与维护。如果网络只需要与时钟同步，而不需要十分遵循某一标准时钟，如 UTC，则可能不需要时间提供者。如果不使用时间提供者，建议您使用 DCE 控制程序 (**dcecp**) **clock set** 命令大约每周手工设置一次时间。

图33显示了一个简单的 LAN 配置。您的 LAN 可能非常大，但本图将模拟您网络中的一部分。

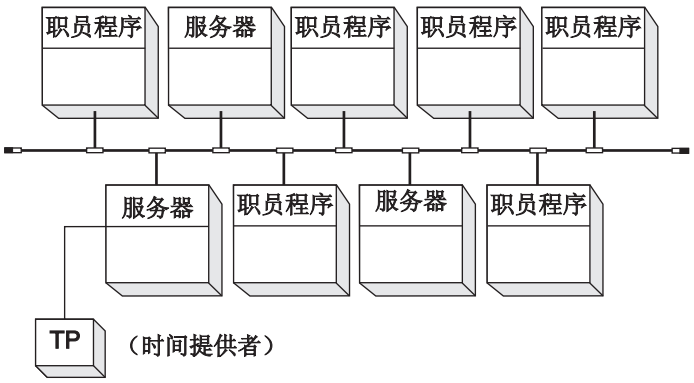


图 33. DTS 配置--LAN

扩展 LAN 的 DTS 配置

如果网络是由若干网桥连接的 LAN 分段构成，则认为这个网络是扩展 LAN。规划扩展 LAN 与规划单一的 LAN 相似；可以把扩展 LAN 的每一分段作为一个独立的 LAN 来对待。建议您参考以下指南:

- 在每个分段中创建三个服务器。
- 使用 **dts configure** 命令将每个分段中的一个服务器配置成全局服务器。
- 使用 **dts modify** 命令将每个分段中的一个服务器配置成信使。
- 如果使用时间提供者，则把它们连到全局服务器。

WAN 以及 WAN 链接的 DTS 配置

由于存在各种各样的 WAN 配置，特别是在 LAN 与扩展 LAN 的组合中，所以不可能描述用于传播时间的 WAN 链接的每一种情况。这一节并没有对每一种涉及 WAN 链接的情况提出建议，但是将描述如何能象示例中那样使用一些类属配置来设置 DTS 实现。

由于在任何 WAN 链接中固有的不定延迟，所以很难维护链路两端的时钟之间差异。DTS 同步整个 WAN 界面内的时钟，但是时钟之间发生了很大的不准确性，原因是在每次同步期间有最坏情况的传输延迟。

可靠的和强壮的 DTS 安装对于是单元一部分的 WAN 链接非常重要。由于 WAN 没有 LAN 可靠，所以应在任何包含 WAN 链接的 DTS 安装中规划一些冗余。请尝试始终保持有三个或更多服务器可用，即使其中的一个 WAN 链接断掉。

以下分段给出了三种推荐的基本配置:

- 用 WAN 链接到远程节点的 LAN 或扩展 LAN
- 由 WAN 链接连接起来的 LAN
- 带有中央主机或群集的全部由 WAN 组成的单元

您的单元可能无法与任何一个配置完全匹配，但是您可以通过以下推荐的每个示例规划单元。

使用到远程站点 WAN 链接的 LAN

图34显示了一个使用 WAN 链接连接若干远程节点的 LAN。

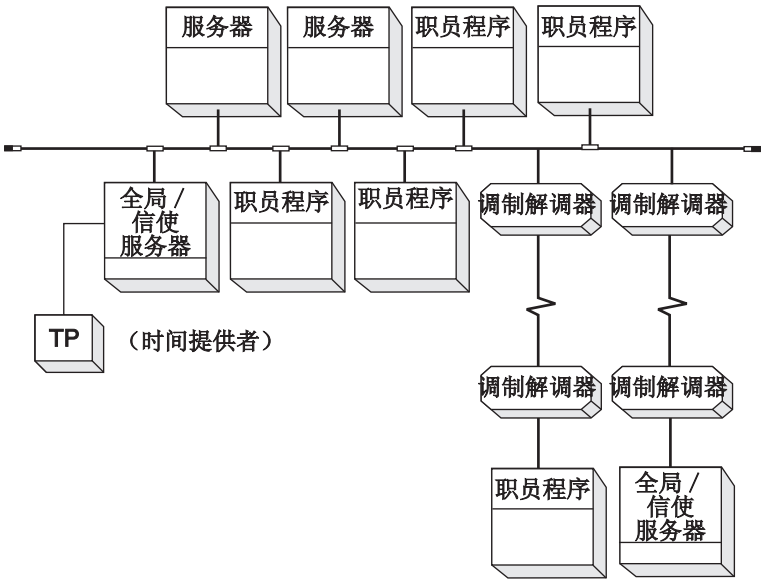


图 34. DTS 配置--使用 WAN 链接的 LAN

在这一配置中，请遵循有关单一 LAN 的基本建议，但是还需坚持以下这些规则:

- 将服务器配置在远程站点上作为全局或信使服务器。
- LAN 中最少应该有三个服务器。
- 如果正在使用单一时间提供者，则把它放在 LAN 上的某一个全局服务器上，而不要放在远程服务器上。

根据上述规则形成的网络配置集中了 LAN 上的服务器；所以，时钟偏差保持在最小并且服务不依赖于那些对系统管理员而言在物理上不可访问的远程节点。每个远程职员程序节点都与全局服务器同步，以符合 **minservers** 属性设置。

由 WAN 链接连接起来的 LAN

用于那些使用网桥的扩展 LAN 的规则纲要也适用于由 WAN 链接连接起来的 LAN。此类网络中的每个 LAN都是一个独立的实体，所以必须为所有的 LAN 配置单独的 DTS 服务器。请根据下列指南配置每个 LAN:

- 在每个 LAN 上至少配置三个 DTS 服务器。
- 在每个 LAN 上至少配置一个服务器作为信使。
- 在每个 LAN 上至少配置一个全局服务器。
- 如果使用时间提供者，应在全局服务器上安装它们。

这些建议能促使 DTS 效率与可用性的提高，尽管存在与 WAN 链接相关的不规则的延迟。

WAN 单元

图35显示了一个在地理上分布的没有任何 LAN 的单元。DTS 在全部是 WAN 的环境中传递的时钟偏差高于全部是 LAN 的环境，但是它仍然提供对于分布式应用程序而言足够的同步。在此类网络中，时钟偏差通常少与 5 秒，但是如果使用了卫星链接，则这个偏差可能会达到 30 秒。

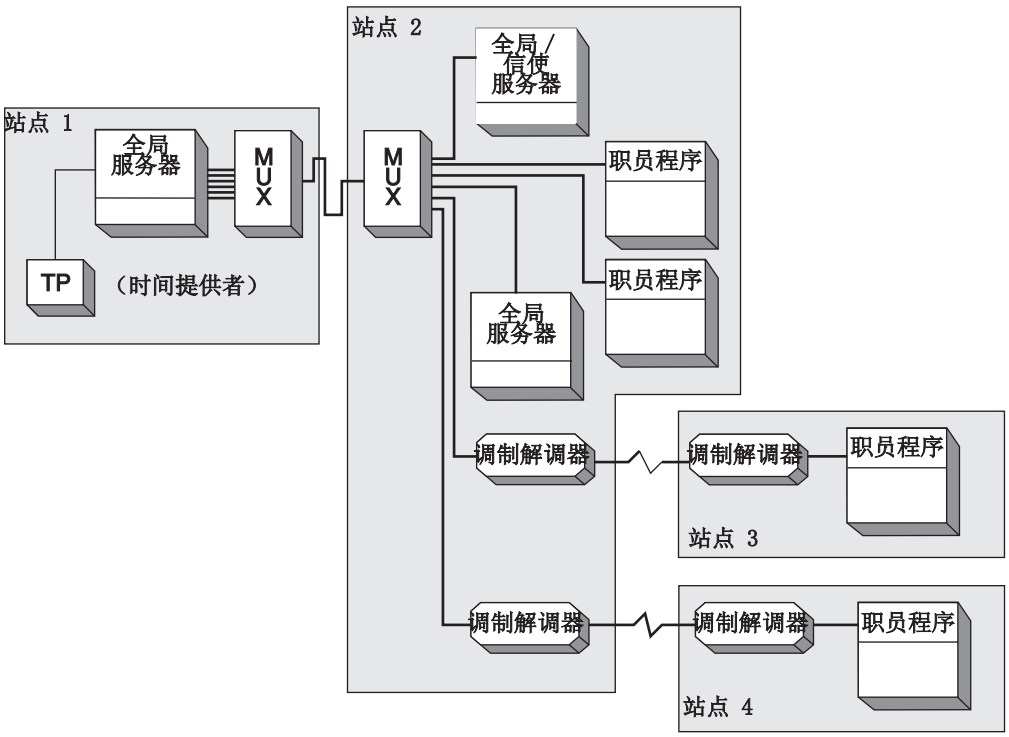


图 35. DTS 配置--WAN 网络

用于使用 WAN 链接的 LAN 的许多相同的建议同样适用于没有任何 LAN 的网络。请在规划全部是 WAN 的网络时，考虑以下内容:

- 网络中应该至少有三个服务器，最好有四台或更多。
- 每个服务器都应该配置成全局服务器。
- 不需要信使；但是您也可以把任何一个或所有服务器配置成信使。**minservers** 属性将迫使每个全局服务器至少与其它两个同步。

- 服务器可以放置在网络中的任何位置，但在中央位置上至少要放置一台；选择一台最活跃的与余下的服务器有着最可靠的链接的远程节点。
- 如果使用时间提供者（建议在这类网络中使用），则把其中一个连接到中央位置上的全局服务器节点。

在一个地理位置上分布的 WAN 中，系统管理员或服务人员也应考虑到访问节点的方便程度。如果要把一台服务器放到通信模式和链接可靠性相同的两个节点中的一个，则应把这台服务器放在中央位置或管理设施易于访问的那个节点。

规划外部时间提供者

要使系统与 UTC 非常同步，可以在网络中放置一个或多个时间提供者。时间提供者有许多形式；它们可以是无线电接收器、软件／调制解调器的组合或是卫星接收器。（有关可以在网络中用于集成这些设备的 “时间提供者接口”的附加信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Application Development Guide--Core Components*。（有关时间源列表，参阅第465页的『附录C. 时间提供者和时间服务』。）

如果计划在网络中使用时间提供者，则可以使用与 *dcelocal*/**usr/examples/dts** 中的 DTS 软件一起提供的样本时间提供者程序中的一个。如果计划使用没有可用样本程序的时间提供者，或有特殊需求，则可以自己编写与时间提供者接口匹配的时间提供者程序。选择了时间提供者设备与程序后，应规划把这个设备安装到网络中的哪个位置。

要把时间提供者放到最有利的位置十分简单。要做到这一点，请遵守以下指南:

- 总是把时间提供者放在服务器上；如果可能，应把时间提供者放在网络中大多数服务器日常可访问到的服务器上。
- 无需考虑网络配置，只要把时间提供者放在最有效与实用的位置即可。
- 如果网络由若干分段构成，并且使用全局服务器来维护整个网络中的同步化，那么，请把时间提供者放在全局服务器上。

注: 不能配置作为信使连接到时间提供者的服务器上。连接到时间提供者的服务器从不假设信使角色，因为服务器只处理来自时间提供者请求的时间值。（有关信使服务器的附加信息，请参阅第207页的『第23章 DCE 分布式时间服务介绍』。）

第25章 管理 DCE DTS

本章描述对 DCE DTS 执行的管理任务。DCE 控制程序 (**dcecp**) 具有可用于执行这些任务的命令。本章包含了对这些命令的概述。*IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference* 中将详细描述这些命令。

在创建 **dcecp** 之前，由 DTS 控制程序 (**dtscp**) 来管理 DTS。您可以一直使用这个控制程序，但其所有操作都已经合并到 **dcecp** 中。而且，您还可以参阅 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference* 以了解有关用 **dtscp** 命令管理 DTS 的详细说明。

使用 DCE 控制程序

因为本指南的第3页的『第1章 DCE 控制程序介绍』描述了有关 **dcecp** 及其命令语法的详细信息，所以本章将不再重复。本章只描述 **dcecp** 提供的特别是用于管理 DTS 的命令。

dcecp 命令用于 DTS 在表示服务组件的对象上执行各种操作。例如，**dts stop** 命令将停止本地节点上的服务器或职员程序。以下各分段将描述 **dcecp** 将在其上进行操作的 DTS 对象以及控制程序可以在这些对象上执行的操作类型。

DTS 对象

DCE 控制程序具有在以下 DTS 对象上操作的功能:

- **dts**

  这个对象表示了以下之一:

  – 本地或全局服务器，它向分布式计算环境中的客户机应用程序和系统提供时间。

  – 中间程序，它扮演了客户机系统上职员程序的角色。DTS 职员程序从 DTS 服务器获取时间并调整时钟。

- **clock**

  这个对象表示本地系统时钟及其告知的时间。

DTS 的 dcecp 操作

表13概述了在 DTS 对象上由 **dcecp** 命令执行的操作。

表 13. DTS 的 dcecp 操作

操作	描述
activate	将职员程序或服务器进程的状态从非活动的更改为活动的并促使对象同步其时间。
catalog	返回指定单元内的 DTS 服务器列表。
compare	比较本地职员程序和指定服务器报告的时间。
configure	将服务器配置成全局或本地服务器。
deactivate	将职员程序或服务器进程的状态从活动的更改为非活动的并让对象停止同步其时间。
help	显示可以在职员程序上执行的操作列表，或对指定对象的详细描述。

表 13. DTS 的 dcecp 操作 (续)

操作	描述
modify	修改职员程序或服务器的属性信息。
operations	显示可以在职员程序、服务器或时钟上执行的简要操作列表。
set	逐渐或立即将时钟设置成由变量（DTS 式样的时间戳格式）指定的时间。
show	对于职员程序或服务器，显示有关属性或计数器的信息。对于时钟，显示用 DTS 式样的时间戳格式表示的时钟时间。
stop	停止职员程序或服务器进程。
synchronize	告诉 **dtsd** 逐步或立即与 DTS 服务器进行同步（**-abruptly** 选项）。

DTS 对象属性和计数器

DTS 职员程序和服务器对象具有属性和计数器，它们是反映或影响它们的操作行为的数据段或数据集。有些 DTS 职员程序和服务器属性由 DTS 守护程序在内部使用，并只允许您查看它们的值（使用 **dcecp dts show** 命令）。其它的则允许您根据环境的需求重新设置值（使用 **dcecp dts modify** 命令）。计数器由 DTS 守护程序在内部使用并包含了您只能查看的值。

表14列出了可以设置的服务器和职员程序属性。表15列出了不可以设置的服务器和职员程序属性。

有 DTS 服务器和职员程序属性与计数器的详细描述，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference*。

表 14. 可设置的 DTS 对象属性

服务器	职员程序
checkinterval	--
courierrole	--
epoch	--
globaltimeout	**globaltimeout**
localtimeout	**localtimeout**
maxinaccuracy	**maxinaccuracy**
minservers	**minservers**
queryattempts	**queryattempts**
serverentry	--
servergroup	--
serverprincipal	--
syncinterval	**syncinterval**
tolerance	**tolerance**

表 15. 不能设置的 DTS 对象属性

服务器	职员程序
actcourierrole	**--**

表 15. 不能设置的 DTS 对象属性 (续)

服务器	职员程序
autotdfchange	**autotdfchange**
clockadjrate	**clockadjrate**
clockresolution	**clockresolution**
globalservers	**globalservers**
lastsync	**--**
localservers	**localservers**
maxdriftrate	**maxdriftrate**
nexttdfchange	**nexttdfchange**
provider	**--**
status	**--**
tdf	**tdf**
timerep	**timerep**
type	**type**
uuid	**uuid**
version	**version**

DTS 时间戳格式

对 **dcecp** 命令的所有响应包含了符合图36中显示的输入与输出格式的时间戳。

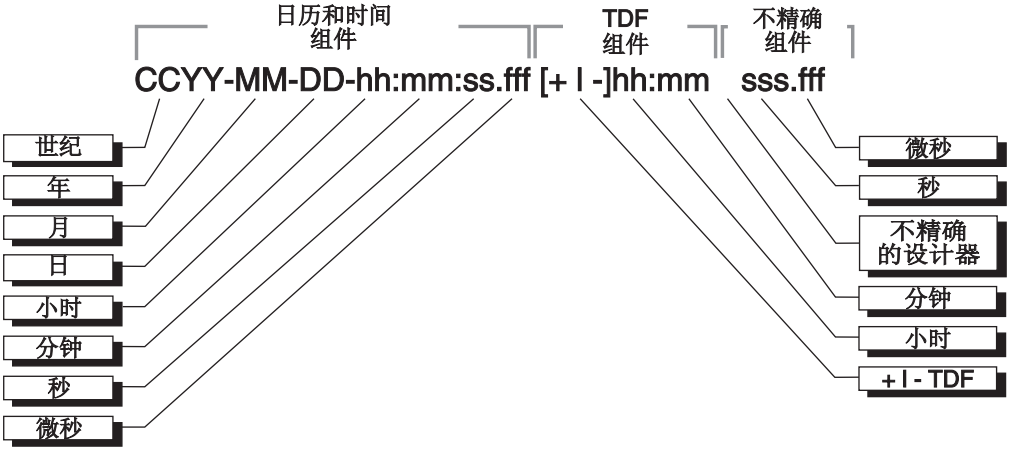


图 36. DTS 时间戳格式

下列示例显示了典型的 DTS 时间屏幕:

```
1994-03-16-14:29:47.52000-05:00I000.003
```

时间戳使用第207页的『第23章 DCE 分布式时间服务介绍』中描述的 DTS 格式。在本示例中，年份是 1994，日期是 3 月 16 日，时间是 14 小时，29 分 47.52 秒。时间戳中包含了一个负的 5 小时的 TDF 以及 3 毫秒的不准确性。

在节点上重新配置 DTS

DTS 最初是在节点的整个 DCE 配置过程期间配置的（参阅 *IBM DCE 版本 3.2 AIX 和 Solaris 版: 管理指南--简介*）。DCE 配置过程自动创建并激活 DTS 服务器以及指定节点上的 DTS 职员程序。但是，您可以在任何时间重新配置节点上的 DTS。如果选择这样做，则必须执行以下步骤:

1. 停止当前正在节点上执行的职员程序或服务器进程（DTS 守护程序）。
2. 运行 **dce_config** 脚本，作为职员程序或服务器来重新启动节点上的 DTS 守护程序。
3. 按需设置任何职员程序或服务器属性值。

以下小节提供了执行列出的每一个重新配置步骤的详细指导。

停止现有的职员程序或服务器

要停止节点上现有的 DTS 职员程序或 DTS 服务器，请使用 **dcecp dts stop** 命令。执行该命令时，首先释放职员程序或服务器（也就是说，禁止职员程序或服务器同步系统时钟的功能），然后停止进程。如下输入 **dts stop** 命令:

```
dcecp> dts stop
dcecp>
```

dts stop 命令调用 **dcecp dts deactivate** 命令释放职员程序或服务器进程。无论何时要释放职员程序或服务器进程都应使用这个命令，但是不停止。请如下输入 **dts deactivate** 命令:

```
dcecp> dts deactivate
dcecp>
```

创建新的职员程序或服务器

要在节点上创建新的职员程序或服务器，请使用 **dce_config** 脚本的这些功能来配置附加的 DTS 职员程序和服务器（请参阅 *IBM DCE 版本 3.2 AIX 和 Solaris 版: 管理指南--简介*）。用于配置附加职员程序和服务器的 **dce_config** 功能重新启动 DCE 守护程序 (**dtsd**) 作为一个职员程序或服务器。

正如在初始的 DTS 配置期间，如果正在创建服务器，则必须告诉 **dce_config** 脚本要创建的服务器类型: 全局或本地。在选择服务器类型前，应考虑服务器将在传播网络时间中扮演的角色。

本地服务器可以扮演非信使的角色（**courierrole** 属性的值被设置成 **noncourier**）。非信使服务器不参与时间传播。本地服务器也可以扮演信使角色（**courierrole** 属性的值设置成 **courier**）或备份信使的角色（**courierrole** 属性的值设置成 **backup**）。信使服务器主要负责在网络片段中的节点之间同步时钟。备份信使是次级链路，它将在没有信使服务器可用的情况下传播时间。在创建本地服务器时，信使角色被自动设置成 **backup**。

全局服务器必须扮演**非信使**角色。不能将它们指定为信使或备份信使。

第237页的『指定全局和信使服务器』 提供了有关服务器信使角色的更详细信息以及在创建服务器后更改信使角色的指导。

设置职员程序和服务器属性值

一旦在节点上创建了新的职员程序或服务器，就需要设置一定的职员属性值。

如果要把节点重新配置成一个服务器，则需要使新建服务器的时段（**epoch** 属性值）与网络片段中先前存在的服务器共享的时段相匹配。要这样做是为了新的服务器能立即与这些服务器同步。第238页的『匹配服务器时段』中给出了更改服务器时段号的指令。 .

您可能还需要检查其余只适用于服务器的属性，使它们成为对先前已存在服务器属性值设置的一种补充。例如，如果服务器含有一个外部时间提供者，则可能需要检查 **checkinterval** 属性。该属性指定了服务器在与 LAN 上的其它服务器同步之前要等待的时间量。

如果因为创建了服务器而打算更改服务器的信使角色，则可以修改 **courierrole** 属性值。

如果创建了职员程序，则可能需要检查新的职员程序的属性值，与网络中先前已存在的职员程序和服务器的属性值进行比较。

第230页的『修改职员程序和服务器属性』中包含了修改 DTS 职员程序和 DTS 服务器属性的常用指令。

临时重新配置 DTS

网络中不时地会发生某个状况或问题，这就要求您在一个或多个节点上临时地重新配置 DTS。LAN 中的节点可能会出现问题，那么您就需要让另一个节点接管该问题节点的职员程序或服务器角色。不要把并非必须的服务器或职员程序添加到网络，而是可以*转换*职员程序或服务器以便让它扮演所需的角色。

如果转换职员程序或服务器，则更改只是临时的。当在节点上停止 DCE 并重新启动时，该节点将回复到它初始的 DTS 配置。最初配置成 DTS 服务器的节点还是服务器；最初配置成 DTS 职员程序的节点仍将是职员程序。要永久更改节点上的 DTS 配置，则必须如第228页的『在节点上重新配置 DTS』中所述运行 **dce_config** 脚本。

要把职员程序临时转换成服务器，或反之，请执行以下步骤:

1. 通过使用 **dcecp dts stop** 命令，停止当前正在节点上执行的职员程序或服务器进程:

   ```
   dcecp> dts stop
   dcecp>
   ```

   停止职员程序或服务器后，退出 **dcecp**。

2. 通过执行带有适当选项（**-c** 选项用于职员程序或 **-s** 选项用于服务器）的 **dtsd** 命令，重新启动节点上作为职员程序或服务器的 DTS 守护程序。例如，要创建本地服务器，请输入以下命令:

   ```
   dtsd -s
   ```

   这个示例命令创建了一个作为备份信使（缺省时，服务器的 **courierrole** 属性值被设置成 **backup**）的本地服务器。如果希望，可以通过使用命令的 **-k** 选项在 **dtsd** 命令行中为服务器指定另一个信使角色。与备份信使不同，示例中创建的本地服务器可以是一个信使 (**courier**) 或不是任何信使角色 (**noncourier**)。

在以下示例中，本地服务器被授予信使角色:

```
dtsd -s -k courier
```

要创建全局服务器，请输入带 **-g** 选项的 **dtsd** 命令:

```
dtsd -s -g
```

注: 如果正在把之前运行 DTS 职员程序的节点重新配置成运行 DTS 服务器，则需要执行额外的步骤。必须在 DCE“安全性服务”注册表中创建主体帐户，并且必须把服务器名添加到现有的 DTS 服务器组 (**dts-entity**)。否则，DTS 职员程序将无法找到新创建的服务器。有关创建主体帐户的指令，请参阅本指南的第311页的『第31章 创建和维护帐户』。有关把主体名称添加的组的指令，请参阅第283页的『第30章 创建和维护主体、组和组织』。

3. 通过使用 **dts modify** 命令，按需设置任何职员程序或服务器的属性值。以下部分提供修改 DTS 职员程序和服务器属性的指令。

修改职员程序和服务器属性

许多管理任务都参与了 DTS 职员程序和 DTS 服务器属性的修改。DCE 控制程序提供了几条显示和更改这些实体属性的命令。

要显示 DTS 职员程序或 DTS 服务器的属性值，请使用 **dts show** 命令。(**dts show** 命令还可用于查看 DTS 实体计数器的值；但不能修改计数器的值。

例如，要显示本地节点上所有客户机和服务器的属性值，请输入以下命令:

```
dcecp> dts show
{checkinterval +0-01:30:00.000I-----}
{epoch 0}
{tolerance +0-00:10:00.000I-----}
{tdf -0-05:00:00.000I-----}
{maxinaccuracy +0-00:00:00.100I-----}
{minservers 3}
{queryattempts 3}
{localtimeout +0-00:00:05.000I-----}
{globaltimeout +0-00:00:15.000I-----}
{syncinterval +0-00:02:00.000I-----}
{type server}
{courierrole backup}
{actcourierrole courier}
{clockadjrate 10000000 nsec/sec}
{maxdriftrate 1000000 nsec/sec}
{clockresolution 10000000 nsec}
{version V1.0.1}
{timerep V1.0.0}
{provider no}
{autotdfchange no}
{nexttdfchange 1994-10-30-06:00:00.000+00:00I0.000}
{serverprincipal hosts/gumby/self}
{serverentry hosts/gumby/dts-entity}
{servergroup subsys/dce/dts-servers}
{status enabled}
{uuid 000013ed-000b-0000-b8ef-03a4fcdf00a4}
dcecp>
```

此屏幕示例显示了位于本地节点的单一服务器的属性值。**dts show** 命令显示的职员程序属性是不同的。而且，屏幕上还将显示服务器的更多属性（请参阅 第226页的表14 和第226页的表15）

如果希望修改 DTS 职员程序或服务器的值，可使用 **dcecp dts modify** 命令。在下面各小节将显示这个命令的若干示例，描述职员程序和服务器的可设置属性。这些小节还将根据网络配置提供对各属性设置的建议。

minservers 属性

minservers 属性指定了在 DTS 可以同步本地时钟前有多少服务器必须向系统提供时间值。

minservers 属性的缺省值和最小建议值为 **3**；系统需要用来自三个服务器的值来计算一个可靠的新时间。根据它是服务器还是职员程序，系统对网络中的其它系统会有不同的需求。

- 职员程序需要来自三个服务器的值。
- 服务器需要来自另两个服务器的值。每个服务器在计算新值时都使用它们自己的时钟。

要重新设置 **minservers** 属性值，请输入带 **-change** 选项的 **dts modify** 命令来设置希望的值。命令接受的值为 **1** 到 **10**。例如，要把要求的服务器数增加到 **4**，请输入以下命令:

```
dcecp> dts modify -change {minservers 4}
dcecp>
```

虽然 **localservers** 属性（指定 LAN 中的本地服务器数）和 **minservers** 属性之间没有直接的关系，但是 **minservers** 属性值通常是所有本地服务器的一个子集。要查看这两个属性的当前值或其中一个值，可以使用 **dts show** 命令。一直等到 LAN 上的 DTS 节点运行至少 10 分钟后，才发出命令。用这种方法，**dts show** 命令确认显示节点同步列表中的所有本地服务器。在输入 **dts show** 命令时可以带选项（-attributes 或 **-all**），也可以不带选项，如:

```
dcecp> dts show
{checkinterval +0-01:30:00.000I-----}
{epoch 0}
{tolerance +0-00:10:00.000I-----}
{tdf -0-05:00:00.000I-----}
{maxinaccuracy +0-00:00:00.100I-----}
{minservers 4}
{queryattempts 3}
{localtimeout +0-00:00:05.000I-----}
{globaltimeout +0-00:00:15.000I-----}
{syncinterval +0-00:02:00.000I-----}
{type server}
{courierrole backup}
{actcourierrole courier}
{clockadjrate 10000000 nsec/sec}
{maxdriftrate 1000000 nsec/sec}
{clockresolution 10000000 nsec}
{version V1.0.1}
{timerep V1.0.0}
{provider no}
{autotdfchange no}
{nexttdfchange 1994-10-30-06:00:00.000+00:00I0.000}
{serverprincipal hosts/gumby/self}
{serverentry hosts/gumby/dts-entity}
{servergroup subsys/dce/dts-servers}
{status enabled}
{uuid 000013ed-000b-0000-b8ef-03a4fcdf00a4}
dcecp>
```

在以前的示例中，**minservers** 属性值设置成 **4**。该设置提供了冗余；在这种情况下，网络中没有全局服务器，即使本地服务器变为不可用，系统仍然同步。

无论何时，系统不能达到 **minservers** 属性设置指定的服务器数，则增加 **toofewservers** 计数器，记录该事件，并显示事件消息检测到的服务器太少。事件消息中包含的信息显示了当前可用的服务器数以及需要的服务器数。如果发现显示了该事件消息，则检查是否有任何服务器发生故障、测试通信链路以确保系统没有从服务器隔离或添加服务器到网络。

可以以其它方式使用 **minservers** 属性，这取决于您的网络配置。请考虑以下情况:

- 如果网络中只有少量的系统，并且希望同步节点时不必考虑服务器的偏差，请把要设置的 **minservers** 属性值降低为 **1** 或 **2**。虽然产生的同步时间是不太可靠的 UTC 度量，但您可以增加系统同步的可能性。但是，如果该设置小于 **3**，系统将无法标识出错服务器。随后的服务器时钟偏差导致了与 UTC 的不同。
- 要增强容错能力并确保系统计算出可靠的时间，请把 **minservers** 属性值设置成 **3**（缺省设置）或更高。然后，系统可以标识出错服务器并计算出它们接收的时间间隔的最小重叠值。但是，请记住，系统将在至少有三个服务器可用时才同步。

网络中的节点数和应用程序类型确定了是保证同步更为重要还是可靠时间与容错更为重要。

全局服务器 minservers 属性的用法

如果网络由一个以上的 LAN 组成，就应该有一组全局服务器。可以通过将本地服务器加入单元概要来创建全局服务器。（更详细信息，请参阅第237页的『宣布全局服务器』。）

网络中存在全局服务器可以影响您选用的 **minservers** 属性的值。如果可用于职员程序或服务器的本地服务器数小于 **minservers** 属性设置，则职员程序或服务器将自动搜索全局服务器名的单元概要。然后，职员程序或服务器从全局和本地服务器请求时间值。

通过输入 **dts show** 命令和查看 **globalservers** 属性值，可以查看并了解全局服务器是否存在。可以如下输入带选项（**-attributes** 或 **-all**）或不带任何选项的 **dts show** 命令:

```
dcecp> dts show
{checkinterval +0-01:30:00.000I-----}
{epoch 0}
{tolerance +0-00:10:00.000I-----}
{tdf -0-05:00:00.000I-----}
{maxinaccuracy +0-00:00:00.100I-----}
{minservers 3}
{queryattempts 3}
{localtimeout +0-00:00:05.000I-----}
{globaltimeout +0-00:00:15.000I-----}
{syncinterval +0-00:02:00.000I-----}
{type server}
{courierrole backup}
{actcourierrole courier}
{clockadjrate 10000000 nsec/sec}
{maxdriftrate 1000000 nsec/sec}
{clockresolution 10000000 nsec}
{version V1.0.1}
{timerep V1.0.0}
{provider no}
```

```
{autotdfchange no}
{nexttdfchange 1994-10-30-06:00:00.000+00:00I0.000}
{serverprincipal hosts/gumby/self}
{serverentry hosts/gumby/dts-entity}
{servergroup subsys/dce/dts-servers}
{status enabled}
{uuid 000013ed-000b-0000-b8ef-03a4fcdf00a4}
dcecp>
```

dts show 显示本地节点知道的所有全局服务器的名称、节点标识和节点名称。

点到点线路上系统 minservers 属性的用法

如果在经由点到点的 WAN 链接连接到 LAN 的系统上使用 DTS，则单一系统将永远不会有一个以上的本地服务器是可用的。建议将此类系统的 **minservers** 属性设置成 **3**。如果将系统配置成职员程序，它将不拥有任何本地服务器并且必须查询三个全局服务器以同步。如果将系统配置成服务器，则必须查询两个全局服务器以同步。

maxinaccuracy 属性

maxinaccuracy 属性指定了 DTS 导致系统同步之前允许的系统的最大不准确性界限。当系统 **maxinaccuracy** 属性设置确定的界限，DTS 不强制系统同步直到不准确性减少到设置的水平或更低。使用 **maxinaccuracy** 属性设置作为用于同步的触发器。可以通过改变设置来改变系统间同步的容许偏差，但是要知道，设置越低，网络负荷越重。缺省设置是 0.10 秒（100 毫秒）。

maxinaccuracy 属性设置对系统同步化行为的作用如下:

- 系统的时钟值比 **maxinaccuracy** 属性值积聚了更多的不准确性，并且 DTS 启动同步。
- DTS 计算新的时间值。
- DTS 调整系统时钟。
- 如果新的时钟设置仍然超过 **maxinaccuracy** 属性值，或如果时钟偏差随后导致达到该值的不准确性，则重复循环。

注意，如果重复后同步仍无法达到小于 **maxinaccuracy** 属性值的不准确性，则系统将继续进行同步。（有关如何让 **syncinterval** 属性避免该循环的信息，请参阅第234页的『syncinterval 属性』。）

缺省的 **maxinaccuracy** 属性值被设计成系统在不打扰网络通信或系统处理的情况下对大多数应用程序保持足够的准确性。如果网络中包括一个或多个能确保极低的不准确性时间提供者，可以降低 **maxinaccuracy** 属性的值。请在下列情况中增加值:

- 如果网络中不使用时间提供者
- 如果系统是仅有 WAN 的网络配置的一个组成部分
- 调用 DTS 的应用程序不需要达到缺省设置的精度度

下列示例显示了如何将 **maxinaccuracy** 属性的值更改成 0.2 秒:

```
dcecp> dts modify -change {maxinaccuracy
00-00:00:00.200}
dcecp>
```

syncinterval 属性

syncinterval 属性避免了使系统同步比指定间隔更频繁。这个属性避免了因 **maxinaccuracy** 属性而一直同步下去。如 第233页的『maxinaccuracy 属性』中提到的，只要系统的不准确性高于指定的值，**maxinaccuracy** 属性触发系统同步。**syncinterval** 属性避免了发生比指定间隔值更频繁的同步。（**syncinterval** 属性值是随机的，它避免了几个系统同时进行同步，并且它是平均值而不是精确值。）

maxinaccuracy 和 **syncinterval** 属性是相互依赖的；当以下这*两*个条件都满足时，系统将自动发生同步:

- 时钟的不准确性等于或大于 **maxinaccuracy** 属性值。
- 自上次同步以来的时间等于或大于 **syncinterval** 属性值（少许随机）。

注意，如果系统达到 **syncinterval** 属性设置但没有达到 **maxinaccuracy** 属性设置，系统不同步。

缺省的 **syncinterval** 属性值是 2 分钟（服务器）和 10 分钟（职员程序）。如果尝试最小化系统之间的偏差，可以降低 **syncinterval** 属性值。例如，如果要让职员程序每 5 分钟同步一次，而且它的不准确性达到了 100 毫秒，请输入以下命令:

```
dcecp> dts modify
-change {syncinterval 00-00:05:00.0000}
dcecp>
```

syncinterval 属性阻止了 **clock synchronize** 命令的执行。可以通过输入该命令在任何时候同步系统。**syncinterval** 属性只影响由 **maxinaccuracy** 属性触发的自动同步。（详细信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference*。）

tolerance 属性

tolerance 属性确定如果系统时钟出错，DTS 如何反应。时钟出错是一个极罕见的条件，但是有些情况会导致时钟出错，包括:

- 时钟硬件中的缺陷，包括时钟偏差大于制造商的规定。
- 时间提供者发生故障。
- 操作系统丢失了硬件时钟的滴答声。
- 系统内存包含的时钟值遭到毁坏。

在同步进程期间，如果时钟值和它的不准确性与用于同步的服务器的那些值不交叉，DTS 将检测到系统时钟出错。第235页的图37中显示了这个过程，其中值 **t2** 是错误的。

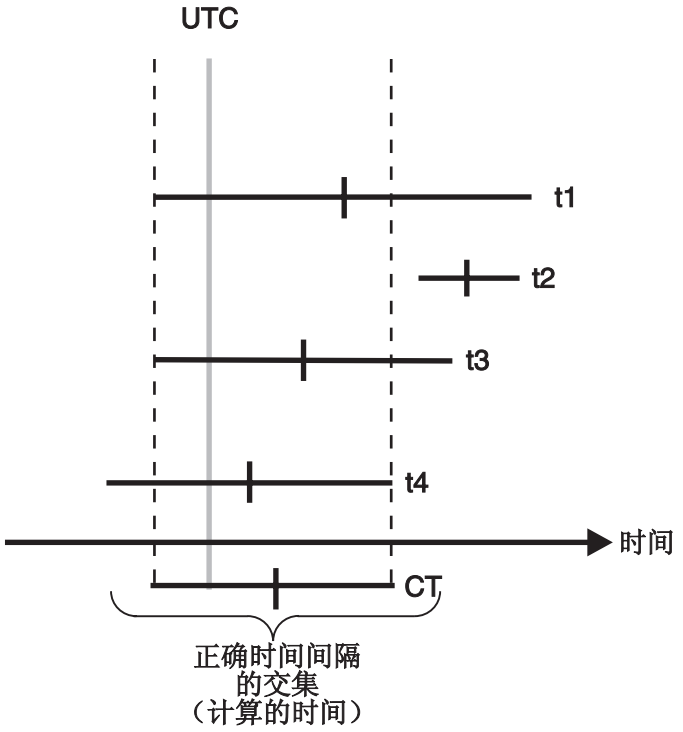


图 37. 本地故障

如果 DTS 在同步期间检测到出错的系统时钟，则出错的严重性和系统的 **tolerance** 属性设置将确定 DTS 如何反应。当检测到故障，DTS 执行以下操作之一:

- 如果由时钟提供的出错时间间隔在容许范围内，则 DTS 增加时钟所提供值的不准确性并逐步调整时钟。
- 如果由时钟提供的出错时间间隔在容许范围外，则 DTS 将立即将时钟设置成计算出的时间。

在更改缺省的 **tolerance** 设置（5 分钟）之前，请确定使用系统时间的应用程序的需求。有些分布式应用程序，如 CDS 服务器，需要系统有不大于 5 分钟的不准确性。较大的容错可能会妨碍此类应用程序正确地排列 CDS 名称空间项的次序。对于这些应用程序，需要把 **tolerance** 属性值设置为 5 分钟或更少。

有些应用程序可能需要 DTS 逐步地并且单调地（向前）调整系统时钟。可以增加这些应用程序的 **tolerance** 属性设置，确保只有在严重错误事件中才会突然设置时钟。如果可以将 **tolerance** 属性值设置成无穷大，则可以保证时钟不会被突然设置。这个设置不可用，但您可以输入任何小于 10675199-00:00:00.000（大约 29,227.5 年）的设置。

下列示例显示了如何把 **tolerance** 属性值设置成 3 分钟:

```
dcecp> dts modify -change {tolerance 00-00:03:00.000}
dcecp>
```

localtimeout、globaltimeout 和 queryattempts 属性

当系统查询服务器时，它在等待响应时将等待由 **localtimeout** 或 **globaltimeout** 属性指定的一段时间。**localtimeout** 属性设置应用在系统尝试联系本地服务器的时候；**globaltimeout**属性设置应用在系统尝试联系全局服务器的时候。

queryattempts 属性确定了当系统不再尝试联系给定服务器之前， DTS 将复位多少次超时定时器。一旦超时设置已经经过了 **queryattempts** 属性设置的次数，则系统将不再查询服务器。如果系统正在查询一个全局服务器，则 DTS 将生成**服务器不响应**事件报告并从系统的全局服务器列表中删除该服务器。如果要求来自全局或本地服务器的响应符合 **minservers** 属性设置，则 DTS 将生成**服务器太少**事件报告，并且系统不同步。

queryattempts 属性的缺省设置是 **3**。以下示例显示了如何把 **queryattempts** 属性值设置成 **4**:

```
dcecp> dts modify -change {queryattempts
4}
dcecp>
```

localtimeout 属性的缺省设置是 5 秒， **globaltimeout** 属性的缺省设置为 15 秒。全局设置较大是为了说明经常用于访问全局集的 WAN 链接上的通信延迟。与此不同的是，您将不得不更改 **localtimeout** 属性设置。但是，**globaltimeout** 属性设置也可能会因为 WAN 的拓扑结构以及传输质量等变化而被更改。在以下示例中，**globaltimeout** 设置被更改成 20 秒:

```
dcecp> dts modify -change {globaltimeout 00-00:00:20.000}
dcecp>
```

如果继续接收到全局服务器的**服务器不响应**事件报告，应增加 **globaltimeout** 的设置。如果增加设置后事件报告仍然继续，则可能是因为至服务器的通信链路有问题。

serverentry 和 serverprincipal 属性

在 DCE 和 DTS 的初始配置期间，将创建一个 DTS 项的名称用以和 CDS 一起使用，并且创建另一个 DTS 名称用以和注册服务一起使用。如果随后想更改服务器的名称，则可以通过更改服务器的两个属性: **serverentry** 属性和 **serverprincipal** 属性来更改。**dcecp** 属性的缺省设置与 DCE 初始配置期间创建的名称相同; *它们是推荐设置*。这一部分描述对这些属性设置的附加注意事项。如果决定更改 **serverentry** 和 **serverprincipal** 属性值，则确保新值是适当的值。如果不是，则 DTS 将发生故障。

serverentry 属性指定了 CDS 项名称，它将在导出服务器绑定时使用。如果更改该属性的设置，则还将在名称空间中修改该项。以下示例命令设置了 **serverentry** 属性值:

```
dcecp> dts modify  -change {serverentry /.:/hosts/cyclops/dts_ref_node}
dcecp>
```

serverprincipal 属性指定了用于认证的服务器的主体名称。如果使用 **dcecp** 来更改名称，则必须在安全性服务注册表中创建匹配的主体名称和帐户。在执行这些时，必须把新的主体名称添加到现有的 DTS 服务器组 (**dts-servers**) 中。机器主体必须是该授权组中的一个成员。有关创建新的主体帐户的更详细信息，请参阅本指南的第311页的『第31章 创建和维护帐户』，有关添加主体名称到现有服务器组的信息，请参阅第283页的『第30章 创建和维护主体、组和组织』。

下列示例设置了 **serverprincipal** 属性:

```
dcecp> dts modify -change {serverprincipal /.:/hosts/ajax/dts_machine}
dcecp>
```

特定于服务器的管理任务

管理 DTS 服务器涉及了一些特殊的任务。这些任务包括:

- 设置服务器时段
- 为服务器分配信使角色
- 将服务器指定为全局服务器
- 设置用于连接到时间提供者的属性

下列小节将描述这些特定于服务器的任务:

指定全局和信使服务器

如果网络含有 WAN 链接或是一个扩展 LAN，则可能需要使用全局和信使服务器来同步网络中各片段中的节点。要同步整个网络中的节点，请把全局角色分配给一些服务器，并把信使角色分配给选定的本地服务器。(有关规划全局和本地信使服务器位置的建议，请参阅第207页的『第23章 DCE 分布式时间服务介绍』 。要分配服务器角色，请遵循以下小节中的指示。

宣布全局服务器

要把服务器分配给服务器全局集，则必须使用 **dcecp dts configure** 命令宣传该服务器。在宣布服务器的同时向服务器 CDS 名称添加绑定信息，并将服务器项添加到单元概要中。由于 CDS 和概要可用于网络中的每个节点，所以 DTS 可以在单元概要中执行查找以获取在 LAN 上无法找到的节点位置。

以下命令示例显示如何把服务器宣布成全局服务器，从而将它注册成使用 CDS 并把它输入单元概要:

```
dcecp> dts configure -global
dcecp>
```

-global 选项指定服务器应配置成全局服务器而不是本地服务器。

要除去对指定为全局服务器的服务器的指定，请如下使用 **dts configure** 命令:

```
dcecp> dts configure -notglobal
dcecp>
```

这个命令取消了对全局服务器的宣布，除去了单元概要中相应的项以及它与 CDS 名称的绑定信息。

为服务器分配信使角色

信使服务器在维护网络各独立部分中系统间同步时扮演了重要角色。信使服务器在每一次同步时至少向一个全局服务器请求时间值。这个过程启用信使服务器将时间从远程系统传播到 LAN 或本地区域，从而使 LAN 与网络中其它所有部分保持同步。

可以如下为服务器分配三种信使角色（**courierrole** 属性）:

- **备份**
- **信使**
- **非信使**

在创建时分配给全局或本地服务器的缺省信使角色为**备份**。

将 **courierrole** 属性设置成 **courier** 以把服务器指定为至网络中其它部分的主要链路。设置成 **backup** 将把服务器指定为至网络其它区域的辅助链路。备份信使只有在 LAN 上没有其它信使时才起作用。

注意，对于备份信使角色，没有重要的处理或开销处罚；您可以将 LAN 上的某一个服务器指定为信使，并把 LAN 上的其它所有服务器指定为备份信使。如果已经把若干服务器配置成了备份信使并且信使不可用，则具有最小编号的 UUID 的备份信使将成为有效信使。

要把信使角色分配给服务器，请输入以下的 **dcecp** 命令:

```
dcecp> dts modify -change {courierrole courier}
dcecp>
```

要把备份信使角色分配给服务器，请输入以下命令:

```
dcecp> dts modify -change {courierrole backup}
dcecp>
```

匹配服务器时段

启动时，服务器的时段号必须与它同步的其它服务器的时段号相匹配。在同步时，服务器忽略来自时段号与它不匹配的服务器的时钟值。

当初次启用 DTS 服务器时，每个服务器的时段号为 0，所以在初始安装时不需要更改时段号。此后，如果添加一个服务器到现有网络，或把职员程序更改成服务器，请确保新的服务器与之前已存在的服务器具有匹配的时段号。输入 DCE 控制程序的 **dts show** 命令可找到服务器的时段号。例如:

```
dts show /.:/hosts/orion/dts-server
```

检查命令返回的属性列表中的服务器的 **epoch** 属性值。如果刚创建的服务器时段与其它服务器的匹配，则新的服务器可立即同步。但是，如果时段不匹配，并且不更改新服务器的时段，那么新服务器将忽略之前已存在的服务器。以下示例显示了如何在启用服务器后更改服务器时段号:

```
dcecp> clock set -abruptly -epoch 0
dcecp>
```

一旦知道服务器是以正确的时段号启动的，就不要更改时段，除非因严重的系统或网络问题破坏了所有服务器时钟值。在大多数的服务器时钟出错这一可能性很小的事件中，请使用 **dts show** 和 **clock set** 命令来隔离有问题的服务器，以便于能执行故障检测并在不影响其余 DTS 应用程序的情况下进行维护。

设置到时间提供者连接的 checkinterval 属性

如果服务器与时间提供者相连，请设置 **checkinterval** 属性。DTS 使用 **checkinterval** 属性定期检查 LAN 上的所有服务器以确保它们与时间提供者保持同步。在经过了 **checkinterval** 属性设置的时间量时，使用时间提供者（TP 服务器）的服务器将执行以下过程:

1. TP 服务器向 LAN 上的所有其它服务器请求时间值。
2. TP 服务器启动同步进程。
3. TP 服务器标识与它自己的时间间隔不重叠的服务器时间间隔。
4. TP 服务器对他检测到的每个出错服务器发出事件消息。

在以上序列中，请注意，TP 服务器在启动同步进程后并没有真正地设置系统时钟。TP 只是运行该进程来检测出错的服务器。DTS 软件假设， TP 服务器上的时间值是最精确可用的，所以，TP 服务器不使用它从其它服务器上收集到的值来更改它的时钟。TP 服务器的功能是作为其它服务器参照的计时器。

为了更快速地通知出错服务器，可以把检查间隔设置为较小的值，但是您要知道，较小的设置会增加网络资源上的负荷。下列示例显示了如何设置 **checkinterval** 属性值:

```
dcecp> dts modify /.:/hosts -change {checkinterval
00-00:00:30.0000]
dcecp>
```

更改系统时间

使用 **dcecp** 命令有三种更改系统时间的方法。以下各小节将描述更改系统时间的原因，然后显示可用于修改时间的和更改系统时钟的命令示例。

单调地更新时间

如果不使用时间提供者，并且网络系统已经运行了一段时间，则可能需要更新某些系统上的时间以和 UTC 或其它外部参考相匹配。当时间提供者不在网络中，系统保持极其同步，但是它们的时钟与接受的时间标准如 UTC 有偏差。

若要修改服务器系统上的时间，使它更精确，请使用 **dcecp clock set** 命令。DTS 同步进程确保了使用命令提供的新时间被传播到了其它网络系统。为了把系统时钟更新为新时间，为系统指定的新时间和不准确性必须形成比当前系统间隔更小的间隔。

为了有效地使用 **clock set** 命令，必须临时访问可信的时间参考。此类参考可以包括由许多标准组织通过收音机或电话发布的时间信号。您也可以使用最近刚验证过其准确性的时钟。（有关 UTC 时间的提供者，请参阅第465页的『附录C. 时间提供者和时间服务』 。

由于 **clock set** 是手工输入的用于修改绝对时间的命令，所以它对于小的不准确性设置的命令不起作用。该命令可达到的可靠的最小不准确性大约是 1 秒。人为错误和处理延迟将共同使较低的设置不可靠。例如，您输入命令和新的时间，然后开始监控参考时间。当察觉该参考到达了期望的时间，您按下“**回车**”以启动命令。您察觉参考标记和您按下“**回车**”并不完全一致。而且，一旦启动了命令，DTS 还需花时间解释并执行该命令。

以下示例显示如何在服务器系统上单调地更新时间；也就是说，如何复位时钟并最终将调整的时间传播到整个网络:

```
dcecp> clock set  1994-10-07-09:30:15.00I01.00
dcecp>
```

如果系统需要与标准如 UTC 同步的程度小于 1 秒，请考虑购买第465页的『附录C. 时间提供者和时间服务』 中列出的时间提供者中的一个，列表中描述的这些时间提供者弥补了传输与处理的延迟，并且可以提供准确性在毫秒级的时间参考。

非单调地更新时间

在突然想设置服务器系统时间时，可使用带 **-abruptly** 选项的 **clock set** 命令。带 **-abruptly** 选项的 **clock set** 命令将立即（非单调地）更改系统时钟设置为指定的时间，而不是逐步（单调地）调整时间。

注: 在突然更改系统时间时应加以注意。对时间的突然调整适用于系统启动时或当系统时钟出错并标识和更正了问题时。将系统时间设置更改成低于系统已知服务器时间间隔将导致 DTS 声明系统在下一同步时出错。

因为 **clock set** 命令通常用于更正时钟的总错误，所以，如果系统和服务器有相同的时段号，则对于系统的已知服务器来说，为给定系统指定的时间可能是错误的。您*可以*防止正在更改时间的系统被声明是错误的。使用带 **-epoch** 和 **-abruptly** 选项的 **clock set** 命令来设置新的时间以使它与其它系统隔离开来。然后，可以更改其它系统的时间与时段直到所有系统再一次共享相同的时段。当发生网络中的大多数服务器都出错这种很少出现的情况时，这个处理很有用。

为了有效地使用 **clock set** 命令，必须临时访问精确的时间参考。此类参考可以包括由许多标准组织通过收音机或电话散发的时间信号。您也可以使用最近刚验证过其准确性的时钟。(有关时间参考源列表，请参阅第465页的『附录C. 时间提供者和时间服务』。)

由于 **clock set** 是手工输入的用于修改绝对时间的命令，所以它对于小的不准确性命令不起作用。该命令可达到的可靠的最小不准确性大约是 1 秒。人为错误和处理延迟将共同使较低的设置不可靠。例如，您输入命令和新的时间，然后开始监控参考。当察觉该参考到达了期望的时间，您按下“**回车**”以启动该命令。您察觉参考标记和您按下“**回车**”并不完全一致。而且，一旦启动了命令，DTS 还需花时间解释并执行该命令。

下列示例显示了如何同时更改系统的时间与时段:

```
dcecp> clock set 1993-10-07-09:30:15.0000I01.0000 -abruptly
-epoch 1
dcecp>
```

强制系统同步

一旦创建并启用了网络中所有系统上的 DTS，那么它们将在没有更多介入的情况下同步。但是还有一些情况是，您希望立即迫使系统同步，而不是在等待由 **syncinterval** 和 **maxinaccuracy** 属性指定的一段时间后才迫使系统同步。例如，您可能希望使用刚加入到网络中的 TP 服务器来同步系统。

要强制同步系统上的时钟，请使用 **dts synchronize** 命令。如果输入不带 **-abruptly** 选项的 **dts synchronize** 命令，时间将逐步被调整。如果输入带 **-abruptly** 选项的 **dts synchronize** 命令，那么，时间将立即被调整。在示例中提出的这种情况中，可能需要使用带 **-abruptly** 选项的命令使时间提供者提供的较小的时间间隔尽快地传播到整个网络。

```
dcecp> dts synchronize -abruptly
dcecp>
```

控制对 DTS 的访问

可以通过使用 DCE“授权服务”访问控制表 （ACL）指定控制对 DTS 对象访问的特权。

代表给定系统上服务器的 DTS 主体是 DTS 的主要访问控制对象。该主体控制了用户和职员程序或服务器进程的访问。DTS 对象在任何 **dcecp** 命令中使用的缺省名称是 **/.:hosts/***hostname***/dts-entity**。

DTS 的 ACL 可以包含适用于主体（用户或进程）或该主体所属授权组的任何类型的 ACL 项。有关 DCE ACL 设施的讨论以及对 ACL 类型和它们的项的描述，请参阅本指南的第261页的『第28章 使用访问控制表』 。

要显示 DTS 服务器主体 ACL 中的 ACL 项，可以使用 **dcecp acl show** 命令。例如:

```
dcecp> acl show /.:/hosts/Detroit2/dts-entity
{unauthenticated r--}
{user hosts/Detroit2/self rwc}
{group subsys/dce/dts-admin rwc}
{any_other r--}
dcecp>
```

要修改 DTS 服务器主体 ACL 中的任何一项，可以使用 **acl modify** 命令。第261页的『第28章 使用访问控制表』给出了使用此命令的指示。

第26章 与“网络时间协议”的互操作

“网络时间协议 (NTP)”是推荐的因特网协议。NTP 同步子网由树型结构图表示，图中节点表示时间服务器，边表示服务器之间的传输路径。树中的根接点被指定为主服务器，它与无线电广播或校准原子时钟同步。其余节点被指定为辅助服务器用以同步其它服务器（主和辅助服务器）。

一个特定服务器与主服务器之间的子网连线数目确定了该服务器在整个结构中的层次；也就是说，连线数越少，层次越低。层次较低的服务器总是比层次较高的服务器准确性更高。所有服务器都有等同的功能，并且能同时作为下一较低层次的客户机，而又作为下一较高层次的服务器来运作。

无论是主服务器还是辅助服务器，通常与同一层次或较低层次上的若干其它服务器一起运行 NTP。选择算法尝试选择最准确和最可靠的一个服务器或一组服务器，通过它们来同步本地时钟。

NTP 和 DTS 都可以用在嵌入式本地网络（即使用路由器、网关和网桥连接的网络）并同时使用广播和点到点传输媒体的大型计算机网络。DTS 和 NTP 可以同步地运行在同一 LAN 中。

下列部分将描述如何将时间给予本地和远程 NTP 时间源以及如何从 NTP 时间源获取时间，并描述如何预防循环。

从 NTP 时间源获取时间

DTS 提供两个样本时间提供者程序:

- **dts_ntp_provider.c**--从 NTP 服务器获取时间，它来自无线电接收器。用户指定 NTP 服务器的名称和不准确性。
- **dts_null_provider.c**--使用在 DTS 服务器上，该服务器上的时钟已经由外部代理，如 NTP 同步。它设置不准确性，但阻止 DTS 设置时间。用户根据与 NTP 的本地实践来设置不准确性。空提供者对于已经含有 NTP 所管理的无线电时钟可能有帮助。使节点带有无线电时钟 DTS 服务器并使用空时间提供者。

从本地 NTP 时间源获取时间

在运行带有时钟和空时间提供者的 NTP 时钟驱动程序的节点上运行 DTS 服务器。以与时间源（例如无线电时钟）一致的方式指定不准确性。其它 DTS 服务器将采用该时间源的时间。在这种情况下，由于系统与时间源相连，所以它是 NTP 的“第一层”服务器。

遵守以下规则和建议:

- 规则--如果它只是子网中的一个时间源（无线电时钟），请确保其它任何 DTS 都没有向 NTP 给出时间。但是，如果存在其它本地时间源，则该限制不适用。
- 规则--如果没有本地时间源，不要运行空时间提供者。
- 建议--使用非常小的轮询率，大约 1 秒。
- 建议--由于 NTP 调用了 **adjtime( )** 系统调用，应了解本地节点将偶然会发生未具体说明的不准确性。

图38显示了带有本地时间源的 DTS 服务器／客户机是如何从 NTP “第一层” 服务器取时间的。

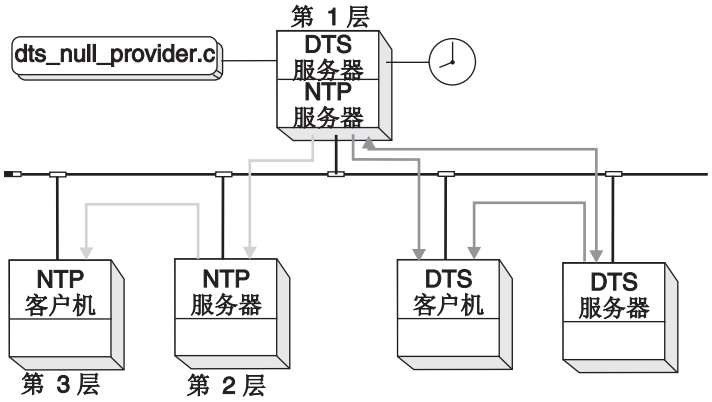


图 38. 本地时间源

从远程 NTP 时间源获取时间

在有权访问 NTP 服务器的节点上运行带有 NTP 时间提供者 (**dts_ntp_provider.c**) 的 DTS 服务器。以与本地 NTP 使用的一致的方式指定不准确性。

遵守以下建议:

- 建议--如果至远程源的链路很远，则应考虑让其中的一个子网节点在本地运行 NTP。
- 建议--注意，NTP 时间提供者不接受来自 “第八层” 或更高层次的 NTP 节点。
- 建议--NTP 节点需要尽可能地靠近 “第一层”。

第245页的图39 和第245页的图40 都显示了 DTS 从远程 NTP 时间源（ “第三层” 服务器）获取时间。但是，在第245页的图39（方案 1）中，应遵守这一节的所有建议；在第245页的图40（方案 2）中，如果至远程源的链路很远，则忽略第一条建议: 让其中一个子网节点在本地运行 NTP。

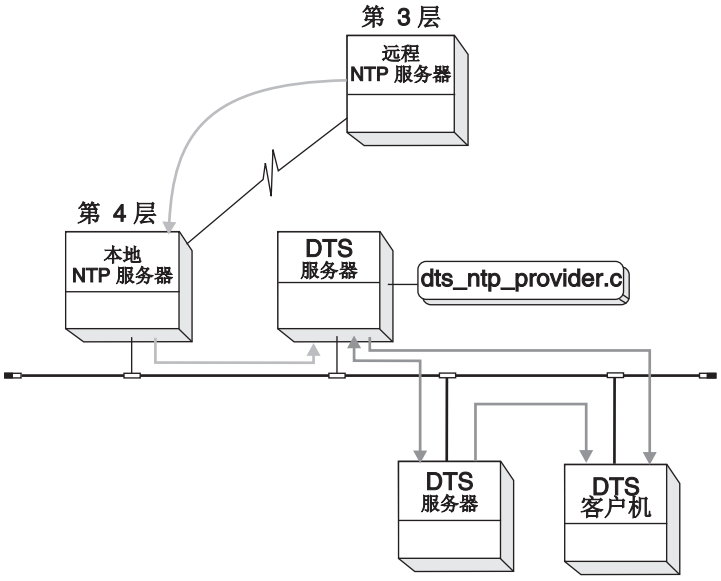


图 39. 从远程 NTP 时间源获取时间（方案 1）

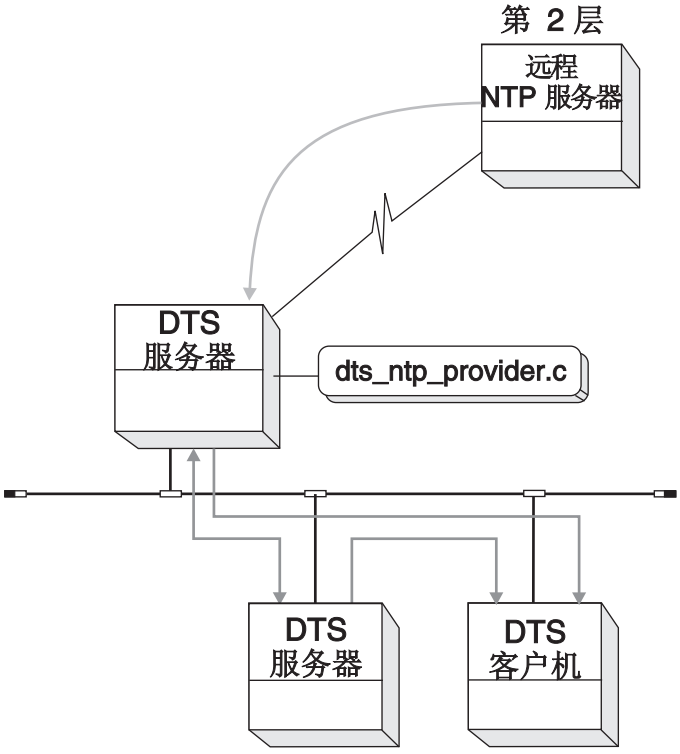


图 40. 从远程 NTP 时间源获取时间（方案 2）

向 NTP 节点提供时间

任何运行带有 **-s** 选项和特殊配置文件 (**ntp.conf**) 的守护程序 **ntpd** 或 **xntpd** 的 DTS 服务器或职员程序都可以作为 NTP 服务器来配置。

对于运行 **ntpd** 守护程序的系统，**ntp.conf** 配置文件必须包含以下这行内容:

```
peer /dev/null DTSS 8 -5 local
```

此外，将 **-s** 添加到文件 **/sbin/init.d/ntpd** 的 **ntpd** 项中，或者，对于使用 **rc.local** 的系统，请相应地修改启动 **ntpd** 的行。

对于运行 **xntpd** 守护程序的系统，**ntp.conf** 配置文件必须包含以下这行内容:

```
peer 127.127.1.8
```

此外，将 **-s** 添加到文件 **/sbin/init.d/xntpd** 的 **xntpd** 项中，或者，对于使用 **rc.local** 的系统，请相应地修改启动 **xntpd** 的行。

在这个配置中，NTP 从不设置时钟。但是，NTP 可以向其它 NTP 客户机提供时间。不允许在 DTS 和 NTP 之间形成循环。如果 NTP 向 DTS 提供时间，那么 DTS 向 NTP 服务器*同一集*返回时间，无法预计的结果会发生。

NTP 配置文件设置成确保从 DTS 获取时间的 NTP 服务器是“第八层”节点。另外，禁止 **dts_ntp_provider** 接受来自“第八层”或更高层次节点的时间。

如果遵守以下规则和建议，则 DTS（服务器）节点可以向 NTP 节点提供时间:

- 规则--**ntp.conf** 文件必须声明该节点在“第 8 层”上。
- 建议--集合中的多个节点可以运行 **ntpd -s** 或 **xntpd -s**。
- 建议--如果任何由 DTS 管理的系统都有本地时间源，那么该系统应作为 NTP **-s** 服务器来使用。
- 建议--虽然这个操作既可以发生在 DTS 服务器上也可以发生在 DTS 客户机节点上，但最好发生在 DTS 服务器上。

注: 如果使用了空供应者，则还必须遵守第243页的『从本地 NTP 时间源获取时间』中的规则，因为运行在 NTP 节点上的空供应者可以忽略层次检查。

第247页的图41显示了两个运行 **ntpd -s** 并向 NTP 子网提供时间的 DTS 服务器。**ntp.conf** 文件将这些服务器定义在“第八层”。

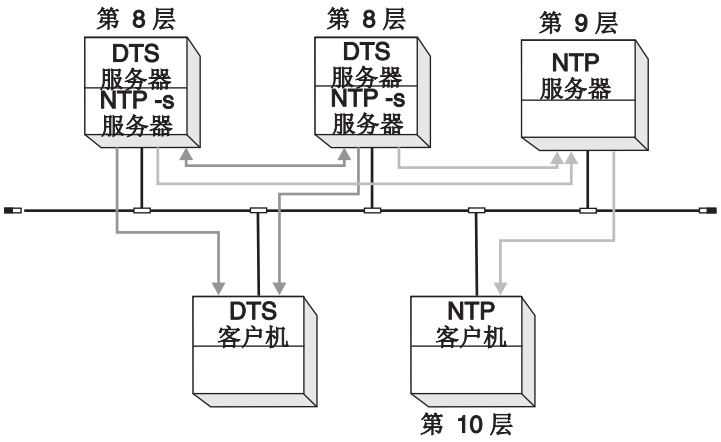


图 41. 向 NTP 提供时间

防止循环

不允许形成循环，例如 NTP → DTS → NTP。

只有在您有本地时间源时才运行空时间提供者 (**dts_null_provider.c**)。如果没有本地时间源，也可以运行空时间提供者，但是*不能*在本地集中任何地方传播 NTP 时间。

图42显示了未建议的配置。只有当远程 NTP“第二层”节点没有发生故障才能使用这一配置。

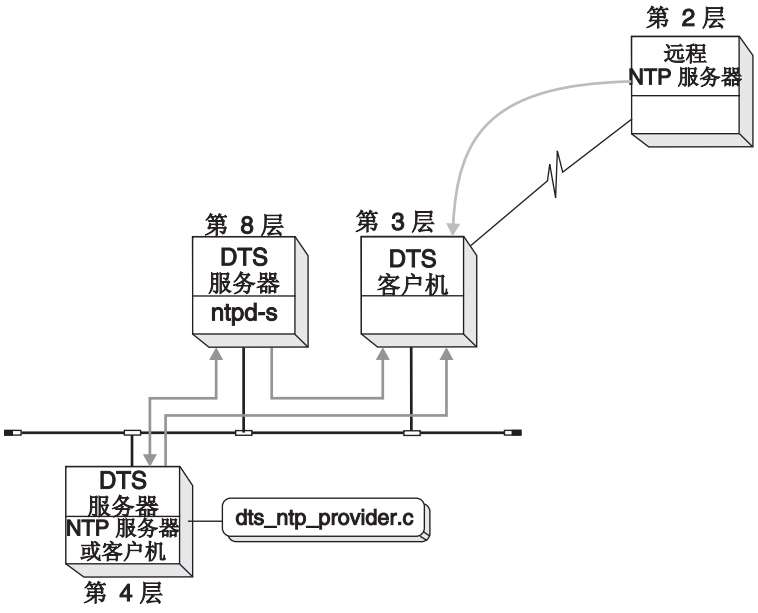


图 42. 第 2 层节点发生故障前的配置

如果远程 NTP 第 2 层节点发生故障，则第 3 层节点开始接受来自第 8 层节点的时间。一旦发生这种情况，第 3 层节点将与第 9 层断开，第 4 层将与第 10 层断开，如第248页的图43中所示。

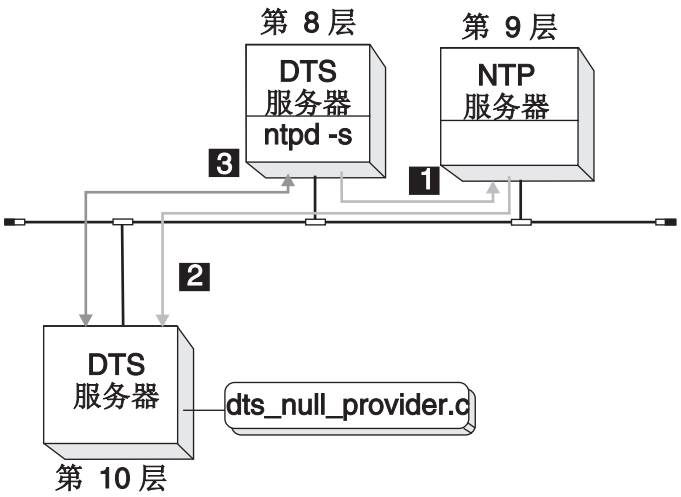


图 43. 第 2 层节点发生故障后的配置

图43中的方案显示了循环的创建:

1. 从标为第 8 层的节点，前进到标为第 9 层的 NTP 节点。
2. 从标为第 9 层的 NTP 节点继续前进到标为第 10 层的节点。
3. 然后，DTS 将时间反馈给标为第 8 层的节点，创建循环。

如果发生循环，NTP 和 DTS 子网中的时间可能与 UTC 有偏差。

第6部分 DCE 安全性服务

第27章 DCE 安全性概述

本章将讨论与“安全性注册表”的非 LDAP 实现（即传统 DCE）相关的内容。有关 DCE“安全性注册表”的 LDAP 实现，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

本章将简单地介绍 DCE 的“安全性服务”。 DCE“安全性服务”由以下服务组成:

- 注册服务 -- 维护注册表数据，它是主体、组、组织、帐户和管理策略的一个复制数据库。
- 认证服务 -- 处理用户认证或验证是否正确标识主体的进程。认证服务还发出主体用于访问远程服务的凭证。该凭证包含请求服务的主体提交给提供服务的主体的数据。
- 特权服务 -- 向用户提供特权属性，用于确保主体有权执行请求的操作。

此外，DCE“安全性服务”还提供了:

- 访问控制列表 (ACL) 设施 -- 根据对象的访问许可权建立并授予对象访问权。
- 扩展注册表属性 (ERA) 设施 -- 提供扩展注册表数据库模式的工具以定义附加属性，并提供将那些属性附加到注册表对象的工具。

DCE 主机守护程序 (**dced**) 充当安全性客户机。

DCE 的“注册表”、“认证”以及“特权服务”都以单一守护程序: (**secd**) 来实现。

DCE 认证服务服务器和客户机

认证服务由注册表数据库、安全性服务器和安全性客户机组成。安全性客户机与安全性服务器 (*dcelocal*/**bin/secd**) 通信以请求信息与操作。安全性服务器访问注册表数据库以执行查询与更新并验证用户登录。要获取注册表数据库的访问权，认证服务必须告诉注册服务。第252页的图44是安全性客户机、服务器以及注册表数据库之间关系的简化表示。

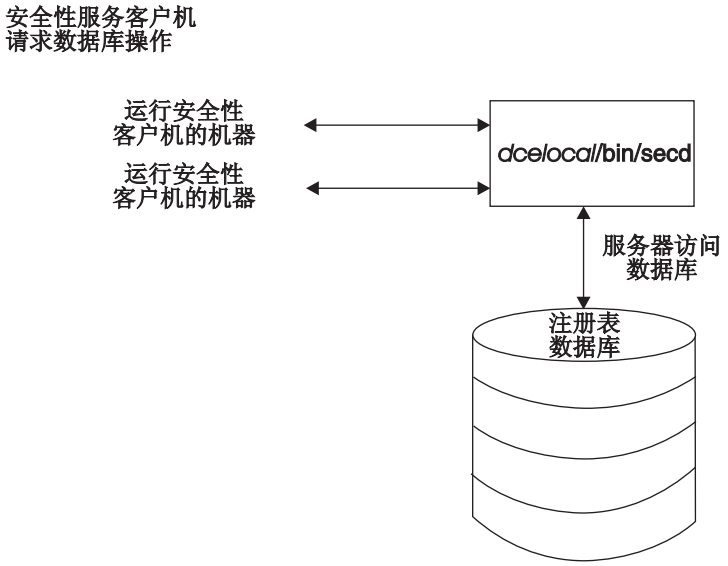


图 44. 机器、服务器和数据库

优选的安全性服务器副本

在 DCE 中，它有时希望客户机在使用复制数据时有个先后次序。对于安全性服务器副本，可以通过在每个客户机上建立 **/opt/dcelocal/etc/security/pe_site** 文件来实现以反映所希望的优先级，并把每个客户机配置成使用此文件找出单元内的安全性服务器。为了有助于该项任务的管理，在 DCE 3.2 AIX 和 Solaris 版中提供了若干增强功能。

1. 称为 **setprofile.tcl** 的脚本与 DCE 客户机一起提供。单元管理员可以使用该脚本将单元内的每个安全性服务器与一个优先级相关联。优选副本的设计依赖于用于存储该信息的 RPC 概要。特别是，脚本将修改客户机指定的 LAN 概要。

   要运行该脚本：从单元内的一个完整客户机，以 **DCE cell_admin** 登录，并在 AIX 或 Solaris 命令行中，运行 interactive，即称为 **setprofile.tcl** 的独立的优先副本。

   它采用客户机的 DCE 主机名以更新为它的唯一变量。该脚本显示可用的安全性服务器副本并允许您交互地建立优先级。允许的优先级为 0-7， 0 表示副本的最高优先级。

   注： 对于单元内使用的每个 LAN 概要只需要运行一次该脚本即可。

   配置期间，为每个 DCE 客户机都分配了一个指向 LAN 概要的指针。要查找为某一特定客户机配置了哪个概要，请运行 dcecp 命令：

   ```
   rpcprofile list/.:/hosts/{dce_hostname}/profile
   ```

   其中 **{dce_hostname}** 表示用于将客户机配置到 DCE 单元的名称。AIX 和 Solaris 的缺省 LAN 概要是 **/.:/lan-profile**。

   要允许客户机组使用指定的安全性服务器优先级设置，请更改缺省的 LAN 概要。例如，它可能希望在地理位置 A 上的一组服务器使用称为 lan-profileA 的概要，该概要给出了位置 A 上的安全性服务器的优先级，而另一位置 B 上的客户机组可能使用了称为 lan-profileB 的概要，该概要给出了位置 B 附近的安全性服务器的优先级。

再次运行 **setprofile.tcl**，或使用 **dcecp rpcprofile remove** 命令从 LAN 概要删除副本的优先级。例如，要从 lan-profile 删除称为 rep1 的安全性副本，请发出:

```
rpcprofile remove -i d46113d0-a848-11cb-b863-08001e046aa5,2.0
                  -m /.:/subsys/dce/sec/rep1
                   /.:/lan-profile
```

在本示例中，**d46113d0-a848-11cb-b863-08001e046aa5,2.0** 表示安全性服务器的接口标识。

2. DCE 主机守护程序 dced 每 24 小时将自动重建 **pe_site** 文件。如果已经建立了安全性服务器优先级，则单元内的每个完整客户机上的 **pe_site** 文件将由 dced 重建以反映这些优先级。如果希望立即更新客户机上的 **pe_site** 文件，请运行 **dcecp secval update** 命令。
3. 缺省时，**TRY_PE_SITE** 环境变量由客户机配置程序设置。在设置该环境变量时，运行在 DCE 客户机上的进程将使用 pe_site 文件中的信息找出单元内的安全性服务器。如果不能与 **pe_site** 文件中列出的副本联系，那么，客户机将尝试使用由 **/.:/sec** 中的 DCE 目录服务维护的安全性服务器列表来查找副本。

注册表数据库

注册表数据库包含以下信息:

- 主体--主体就是指系统用户。主体可是交互式主体（人），也可以是非交互式主体（服务器、机器和单元）。主体可以与访问许可权关联。
- 组--组是指由组名标识的主体的集合。组可以与访问许可权关联。
- 组织--组织是指主体的集合；这些主体由组织名标识。组织定义了与注册表中的主体关联的策略。组织不能与访问许可权关联。
- 帐户--帐户包含了口令与帐户信息，它允许主体对单元内的对象进行经认证的访问。（经认证的访问也可以发生在不同单元内的主体之间，如以下文本所描述的。）
- 策略和特性--策略和特性给定了如口令长度与格式以及一定的认证需求等内容。
- **replist** 对象--此对象用于管理注册表数据库的副本。
- **xattrschema** 对象--此对象是使用 ERA 设施创建的扩展注册表。

（有关注册表数据库的结构以及它所包含的信息类型的详细说明，请参阅第419页的『第43章 访问注册表对象』 。）

受注册表数据库控制的对象集合是一个实体，称为*单元*。 只有当那些单元在它们所希望通信单元的注册表数据库上拥有特殊帐户时，这些单元之间才可能进行通信。这些特殊帐户设置了整个单元的认证，为主体提供了从一个单元到另一个单元的经认证的访问。（有关建立整个单元认证的帐户的信息，请参阅 第339页的『第33章 管理多单元环境』 。）

数据库的物理安全性

注: 在这一主题中，与 DCE “安全性注册表” 以及 LDAP “集成” 功能相关的信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

DCE“安全性服务”为网络安全提供了安全措施，它通过确保在机器之间进行通信的主体的身份合法从而来保护在整个网络中发送的信息。但是，它提供服务的安全性服务器和数据库在一个本地机器上。注册表数据库的安全程度与它所在机器提供的安全性一样。除了要确保只有 root 才能访问本地机器上的敏感数据之外，还需要为安全性服务器所在的机器提供物理安全性。这包括将机器放在一个加锁的房间内，保留何时和由谁访问机器的日志，以及其它任何适合您需要的方法。

注: 有关认证的更详细的讨论，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Application Development Guide--Core Components*。

如何存储注册表数据库

注: 在这一主题中，与 DCE“安全性注册表”以及 LDAP“集成”功能相关的信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

每个安全性服务器都在虚拟内存中维护着一个注册表数据库的工作副本并在磁盘上维护着一个永久副本。对副本的所有读和更新操作都在虚拟内存中完成。当服务器启动时，它使用磁盘上的副本来初始化虚拟内存中的副本。一个更新日志的基本单元用于保证服务器发生故障时数据库的状态。

图45显示了服务器以及注册表数据库的磁盘副本和虚拟内存副本。

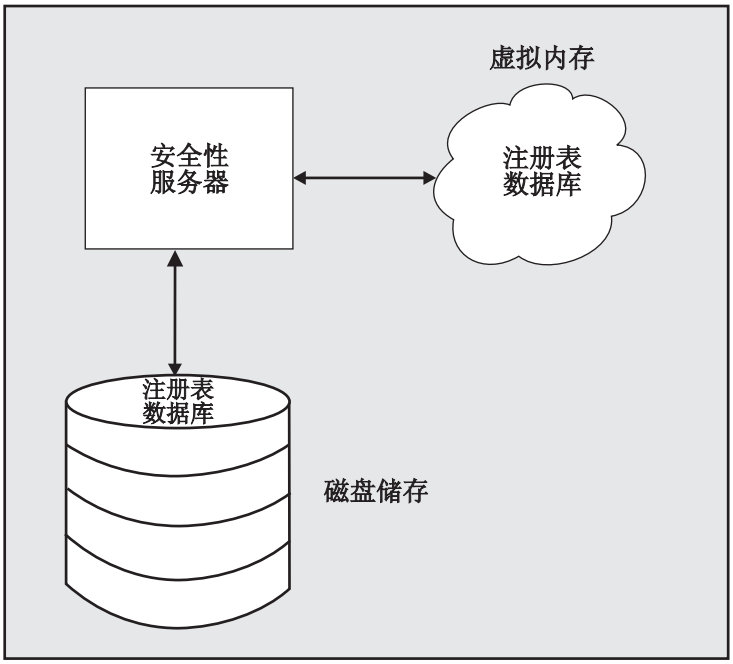


图 45. 注册表数据库的磁盘存储器副本和虚拟内存副本

每个安全性服务器都定期地把虚拟内存上的整个数据库保存到磁盘上。该数据库存储在 *dcelocal*/**var/security/rgy_data** 中。

复制的数据库

注册表数据库可以在其单元内进行复制。单元内的每个安全性服务器都维护着该数据库的一个工作副本。在整本指南中，安全性服务器及其数据（注册表数据库）的组合即称为*副本*。通常，在单元内创建若干副本以增强性能和可靠性。

安全性服务器将自动处理保持单元副本一致的任务。您所作的任何更改都将自动地反映到所有副本中，如以下部分所述。

如何处理更新

注: 在这一主题中，与 DCE “安全性注册表” 以及 LDAP “集成” 功能相关的信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

更新只在一个数据库进行，而所作的更改将传播到所有其它数据库。当传播暂挂时，所有副本都可以访问，即使它们还没有完全更新。换句话说，即使还没有对副本进行更改，副本仍然是可用的。这种复制机制确保了所有副本一直可用于登录验证和读操作，即使更改还在传播过程中。

主和从属副本

单元内只有一个副本（*主副本*）才接受来自客户机对数据库的更新。其它副本，称为*从属副本*，只接受来自客户机的读操作。主副本向从属副本传播任何更新。例如，无论是主副本还是从属副本都可向客户机程序如 **/bin/login** 提供帐户信息。但是，如果要添加一个帐户或要更改口令，则只能由主副本才能处理这些更新。

主副本的更新数据库过程与从属副本的更新过程稍有不同。第256页的图46 和第257页的图47 阐述了主和从属副本的更新过程。这一节中按图示描述了这些过程。

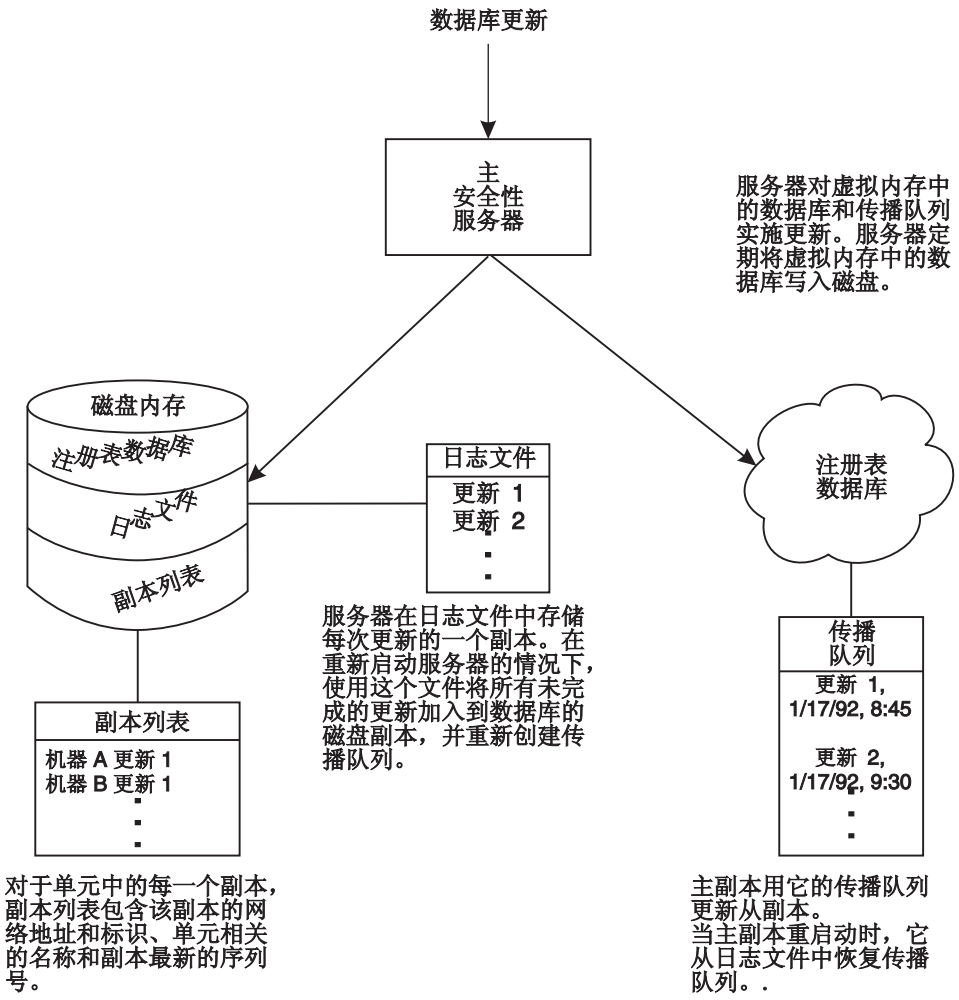

数据库更新
主
安全性
服务器
服务器对虚拟内存中的数据库和传播队列实施更新。服务器定期将虚拟内存中的数据库写入磁盘。
磁盘内存
注册表数据库
日志文件
副本列表
日志文件
更新 1
更新 2
服务器在日志文件中存储每次更新的一个副本。在重新启动服务器的情况下，使用这个文件将所有未完成的更新加入到数据库的磁盘副本，并重新创建传播队列。
注册表
数据库
传播
队列
更新 1,
1/17/92, 8:45
更新 2,
1/17/92, 9:30
副本列表
机器 A 更新 1
机器 B 更新 1
对于单元中的每一个副本，副本列表包含该副本的网络地址和标识、单元相关的名称和副本最新的序列号。
主副本用它的传播队列更新从副本。
当主副本重启动时，它从日志文件中恢复传播队列。.

图 46. 主副本更新过程

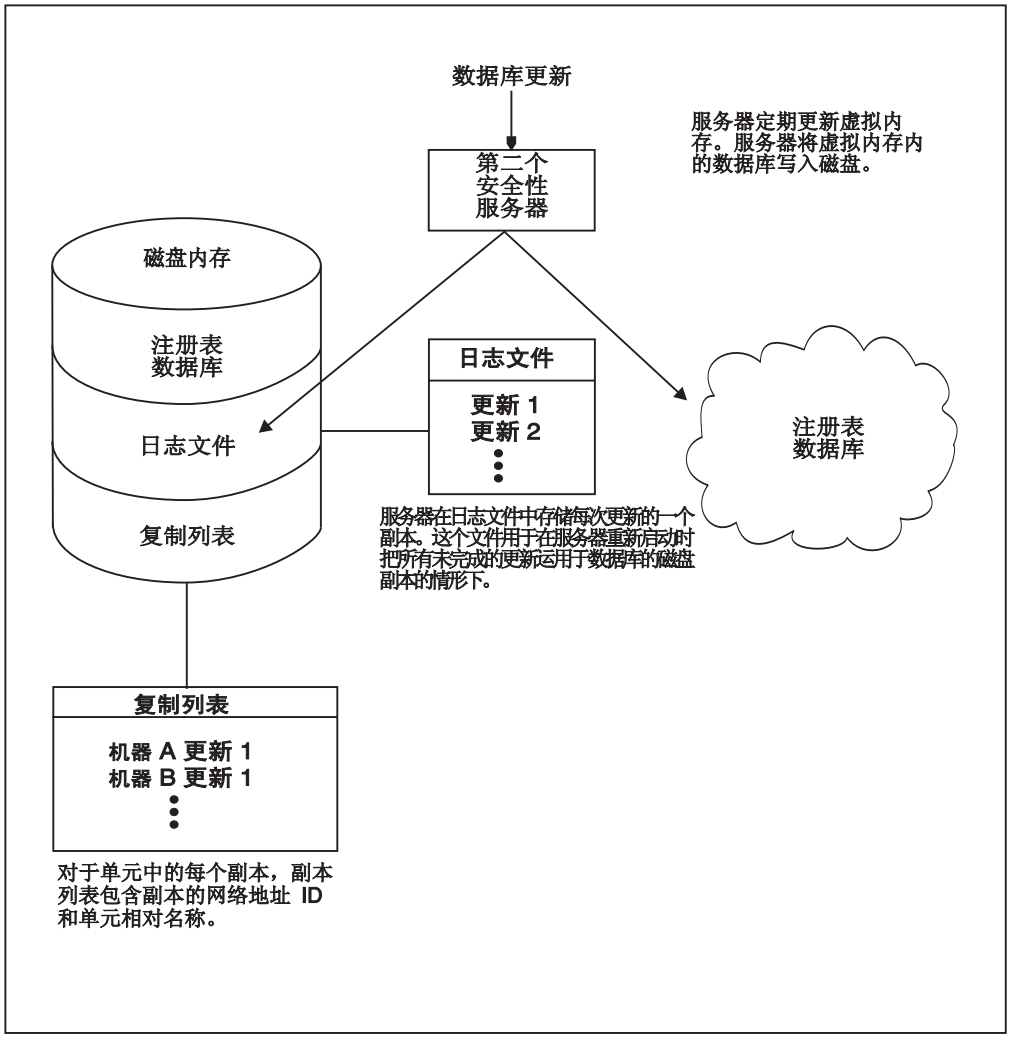


图 47. 从属副本更新过程

处理数据库更新

注: 在这一主题中，与 DCE “安全性注册表” 以及 LDAP “集成” 功能相关的信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

当主副本或从属副本接收更新时，它将这些更新应用到虚拟内存中的数据库，并将每个更新的拷贝保存到磁盘上存储的日志文件中。更新在日志文件中按照序号进行累加。如果服务器意外地重新启动，日志文件必须确保没有丢失任何更新。

副本将定期地将虚拟内存中的数据库写入磁盘，这样磁盘中的副本就得到了更新。然后，如果该副本是从属副本，它将清除日志文件中的所有更新。如果该副本是主副本，它将清除日志文件中已经传播给从属副本的所有更新。还没有传播给从属副本的更新将被保留并在必要时，重新构建传播队列。

只有主副本才拥有传播队列，用以保留要传播给从属副本的更改，如『传播数据库更改』中所述。当主副本接收更新时，它除了把更新添加到其虚拟内存数据库和日志文件外还将把更新添加到传播队列。传播队列中的每个更新都由序号和时间戳标识。序号在内部使用以跟踪对从属副本更新的传播。时间戳被用于向用户显示更新的日期和时间。

当主副本或从属副本重新启动时，它将初始化虚拟内存中的数据库，然后把日志文件中的任何未完成的更新应用到它的数据库中。如果该副本是主副本，则它还将从日志文件重新创建传播队列，这样，就可以传播任何未完成的从属更新。这个机制确保服务器在关机时不会丢失任何更新。

传播数据库更改

注: 在这一主题中，与 DCE "安全性注册表" 以及 LDAP "集成" 功能相关的信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

要将更新传播到从属副本，主副本首先应使用第257页的『处理数据库更新』中描述的过程更新其数据库拷贝。 然后，主副本尝试将更新传播到其副本列表上的每一个从属副本。副本列表中包含了每个从属副本的标识与网络地址。它还包含了对从属副本所作的最后一次更新的序号。主副本总是以序号的先后次序传播。通过检查副本列表中与副本关联的序号以及在传播队列中的更新的序号，主副本可以确定传播队列上的哪个更新必须传播到哪个从属副本上。这个机制有助于确保单一从属副本的无法使用不会妨碍对其余从属副本的更新。

如果首次尝试传播更新不成功，则主副本将定期进行尝试，直到成功为止。当更新成功时，主副本将更新与副本列表上已更新副本相关联的序号。当更新传播到所有从属副本时，主副本将从传播队列中删除该更新。

主 / 从属认证

与所有 DCE 对象相似，主和从属副本必须彼此进行认证。要做到这一点，主副本将传送 **dce-rgy**（在至数据库时创建的主体）的标识。从属副本将传送它所在的主机机器的标识。注意，该标识必须有权访问第419页的『第43章 访问注册表对象』所描述的 **/.:/sec/replist** 对象。

用户至用户的认证

要求在应用程序中使用用户至用户的认证功能，这些应用程序必须能接收受保护的 RPC，但无权访问长期密钥，如密钥表文件。

使用消息传递模型的客户机 / 服务器应用程序需要这一功能。客户机使用这一模型向服务器发出 RPC 以请求一些活动。服务器将接收到的请求排入队列并立即返回给客户机。然后，客户机等待服务器完成该请求并回调。在此方案中，服务器启动受保护的 RPC 用于回调，而客户机扮演服务器的角色。

/etc/passwd 和 /etc/group 文件以及注册表

您在本地机器上必须拥有文件 **/etc/passwd** 和 **/etc/group** 的标准版本以确保与 UNIX 程序的兼容性。要使文件 **/etc/passwd** 和 **/etc/group** 与注册表数据库保持一致，请使用 **passwd_export** 命令。建议经常地运行 **passwd_export**，最好使用 **cron**。（有关 **passwd_export** 的详细内容，请参阅第361页的『第36章 执行例行维护』 。）

注: 与 UNIX 行为不同，如果安全性服务器不可用，则不能使用 **/etc/passwd** 和 **/etc/group** 文件进行本地登录。应使用本地注册表（在以后部分中描述）代替它。**/etc/passwd** 和 **/etc/group** 文件的存在只是为了与需要它们存在的 UNIX 程序相兼容。

本地注册表

每个本地机器的 *dcelocal***/var/security** 目录中的本地注册表都包含了有关最近使用机器的用户以及他们最后一次登录的日期与时间的信息。如果安全性服务器不能用于网络登录，则认证服务将尝试从本地注册表获取本地登录所需的信息。

当安全性服务器运行在网络上时，当任何用户首次从机器上登录到 DCE 时，认证服务将自动创建一个本地注册表。此后，它将在任何用户每次从机器上登录到 DCE 时更新该本地注册表。可以使用带 **-l** 标志的 **rgy_edit** 命令编辑本地注册表。注意，**dcecp** 不访问本地注册表。

安全性对象的名称

因为安全性名称空间来源于“单元目录服务”(CDS) 名称空间，所以安全性对象具有 CDS 路径名，采用以下格式:

/.../cellname/mount_point/object_name

其中:

cellname
表示对象所在单元的名称。

mount_point
表示 DCE“安全性服务”注册到 CDS 时使用的名称。

object_name
创建对象时指定的注册表对象的名称。如果对象在某一目录中，则 *object_name* 由对象本身的名称以及访问该对象必须遍历到的任何目录组成。注意，注册表对象通常在注册表数据库中的主体、组或组织目录中。有关注册表数据库结构的更完整的描述，请参阅第419页的『第43章 访问注册表对象』 。

例如，驻留在单元 **dresden.com** 中的主体 **bach** 的完整路径名使用 **sec**（安全性）安装点并在如下的 **principal** 目录中:

```
/.../dresden.com/sec/principal/bach
```

另一个示例，假设组 **east-west** 在 **sales** 中， **sales** 是 **dresden.com** 单元的注册表数据库中的目录 **group** 的子目录。则 **east-west** 的完整路径名如下:

```
/.../dresden.com/sec/group/sales/east-west
```

dcecp 安全性命令中名称的使用

对于用于管理 DCE“安全性服务”的所有 **dcecp** 命令， **dcecp acl** 除外，您只需提供一个对象名来表示想要操作的对象。对象名存储在注册表数据库中。不需要输入单元名称（假设是本地单元）或安装点（假设是为 DCE“安全性服务”注册的名称）。

dcecp acl 命令中名称的使用

与其它 **dcecp** 安全性命令不同，**dcecp acl** 命令使用由 DCE 服务维护的 ACL而不是使用安全性。与任何在可以存在于不同名称空间中的对象上操作的类属工具相似，**dcecp acl** 需要对象的全限定 CDS 路径名，而不单单是 *object_name*。

例如，要使用 **dcecp acl** 命令更改与主体 **bach** 的注册表帐户关联的 ACL，必须输入以下这个全限定名:

```
/.../dresden.com/sec/principal/bach
```

或

```
/.:/sec/principal/bach
```

还需注意，要使用 **dcecp acl** 操纵注册表数据库中主体目录上的 ACL，并控制谁可以添加或删除主体，则必须输入下列全限定名:

```
/.../dresden.com/sec/principal
```

此单元名还可以表示目录名，例如:

```
/.:/sec/principal/vienna.com/violinists_cell
```

在这些情况下，**dcecp acl** 命令将提供一个选项，以标识是输入目录名还是主体名。

第28章 使用访问控制表

有关“DCE 安全性注册表”和“LDAP 集成”功能的主题信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

可以通过使用 ACLL 授权机制控制对 DCE 对象的访问。 ACL 与文件、目录、CDS 项和注册表对象关联。还可以由任意应用程序实现对内部数据对象的控制访问。每个 ACL 包含多个 ACL 项，定义由谁对对象授权什么操作，具体如下:

- 谁能访问对象
- 主体或组对对象具有什么类型的访问
- 对未经认证的用户允许什么访问

有关特定 DCE 组件如何实现 ACL 授权机制的详细信息，请参阅本指南的适当部分。

注: 在本章及随后的章节中有关 DCE 授权的讨论中，术语 *user* 指主体。主体可以是用户、服务器或机器。

授权概述

ACL 包含项列表，指定可以访问对象的主体和由主体执行的操作。主体可以显式命名或直接是 ACL 项中标识的组成员。ACL 与它保护的对象关联。主体可以执行的操作由许可权指定。

可以为下列项设置 DCE 许可权:

- 所有者、组和其它
- 本地单元和外部单元中特定的个别主体
- 本地单元和外部单元中特定的个别组
- 其中未设置个别许可权的特定外部单元中任何其它主体
- 由 DCE 认证服务认证过的任何单元中的任何主体
- 本地或外部单元中的代理用户、服务器或组
- 未授权的用户

ACL 还为与当前版本不同的 DCE 版本集成保护提供了掩码功能和方法。

文件系统通常设计成为文件系统对象（例如文件和目录）提供访问许可权。DCE 中的 ACL 意义更为广阔。在 DCE 中，许多对象可以具有 ACL 并被赋予许可权。DCE ACL 控制对 DCE 组件管理的对象的访问，如“分布式文件服务”、“DCE 安全性服务”和“DCE 目录服务”。

例如，安全性服务 ACL（控制帐户的组件）能授权特定主体更改与帐户关联的所有信息、授权其它主体仅更改与帐户关联的信息的子集，并限制其它主体更改与帐户关联的任何信息。

DCE 能支持与特定类型对象相应的特定许可权集合。例如，对于容器存在其它对象（例如主体）不需要的插入许可权。这种 ACL 的扩展用法与 POSIX 系统的用法形成对照，

例如，在后者中，仅文件系统对象受许可权位保护，使用许可权的标准集合（读、写和执行）。DCE 控制程序具有命令 **acl permissions**，显示特定于与命名对象关联的 ACL 的许可权。

ACL 管理器

ACL 管理器是处理 ACL 的服务器部分。一个 ACL 管理器能支持几种不同类型的 ACL。从更抽象的观点来看，每个 ACL 类型都是由相应的 ACL 管理器类型支持的。在不正规的情况下，ACL 管理器类型有时被称为 ACL 管理器。图48显示服务器中的 ACL 管理器。

客户机方允许您与导出 ACL 接口的任何服务器连接，以便使用一个程序就能操纵所有的 ACL。DCE 控制程序使用此功能。

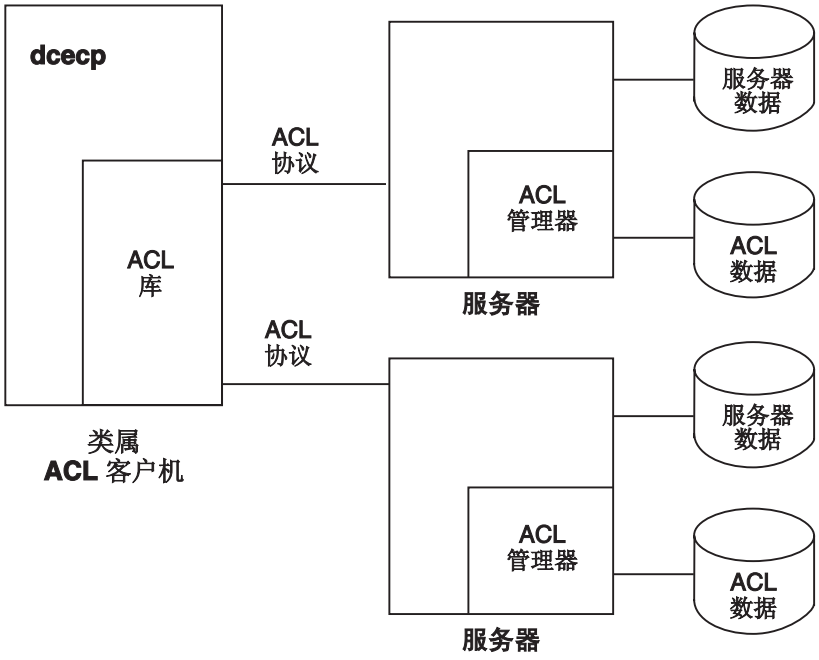


图 48. 服务器中的 ACL 管理器

除了标准 DCE 组件，ACL 还控制对 ACL 管理器已实现的任何对象的访问。ACL 可以与用户编写的应用程序关联以保护对使用应用程序本身、应用程序中的文件甚至那些文件中的字段的访问。

本章中描述的 ACL 的所有元素都对 ACL 管理器可用；但是，每个管理器能实现所有元素或仅元素的一个子集。有关特定 DCE 组件如何使用 ACL 的信息，请查询本指南中适当章节。

在应用程序能在 DCE 更新版本上运行之前，已从与 OSF DCE 发行版 1.0.3 一起提供的示例代码建立 ACL 管理器的用户需要对 daclmgr.h 文件和使用 ACL 管理器的应用程序进行更改。

1. 更改 daclmgr.h：在 API 原型-**sec_acl_mgr_get_access** 和 **sec_acl_mgr_is_authorized**-中，将数据类型 `rpc_authz_cred_handle_t` 替换为 `sec_id_pac_t`，如下所示：

```
extern void DCEAPI sec_acl_mgr_get_access(
#ifdf IDL_PROTOTYPES
/* in */ sec_acl_mgr_handle_t sec_acl_mgr,
/* in */ rpc_authz_cred_handle_t *accessor_info, <-This line becomes
/* in */ sec_id_pac_t *accessor_info, <-this line.
/* in */ sec_acl_key_t_sec_acl_key,
/* in */ uuid_t *manager_type,
/* in */ sec_id_t *user_obj,
/* in */ sec_id_t *group_obj,
/* out */ sec_acl_permset_t *net_rights,
#endif
extern boolean32 DCEAPI sec_acl_mgr_is_authorized(
#ifdef IDL_PROTOTYPES
/* in */ sec_acl_mgr_handle_t sec_acl_mgr,
/* in */ sec_acl_permset_t desired_access,
/* in */ rpc_authz_cred_handle_t *accessor_info, <-This line becomes
/* in */ sec_id_pac_t *accessor_info, <--this line.
/* in */ sec_acl_key_t sec_acl_key,
/* in */ uuid_t *manager_type,
/* in */ sec_id_t *user_obj,
/* in */ sec_id_t *group_obj,
/* out */ error_status_t *st
#endif
```

2. 在应用程序中，将变量 rdaclif_v0_0_epv_t 更改为 rdaclif_v1_0_epv_t。
3. 在应用程序中，将变量 rdaclif_v0_0_s_ifspec 更改为 rdaclif_v1_0_s_ifspec。

ACL 解释

与帐户关联的信息部分是主体和组集。（由于它的 Multics 起始地址，组在此上下文中称为*项目列表*。）与帐户关联的主体和项目列表一起被称为*特权属性*（或客户机方访问控制信息）。

主体和每个组都由字符串名称和 UUID 表示。特权属性 UUID 包含在用于已认证的远程过程调用 (RPC) 或 GSSAPI 调用的凭证中。服务器根据接收到的凭证内容授权访问。虽然服务器一般拒绝未认证的调用（使用无凭证调用），任何服务器仍能支持接受它们的策略。在这种情况中，服务器的 ACL 管理器必须支持**未认证**掩码 ACL 项类型，以便服务器能进一步限制对这种未认证客户机赋予的访问。

当主体请求访问与 ACL 关联的 DCE 对象时，对象的 ACL 管理器将主体和主体属于的任何组（主体的特权属性）的 UUID 与 ACL 项中列出的主体和组的 UUID 相比较。它简单地通过 ACL 项列表读取来实现。管理器在它找到的第一个与任何主体的特权属性匹配的 ACL 项（或组项）中授予访问许可权。如果匹配项中的许可权允许所请求的访问方式，主体将获得访问权；如果不允许，访问将被拒绝。

由进程继承的凭证

主体创建或产生的进程继承主体的凭证。例如，如果登录已认证，且启动了应用程序，则启动的应用程序将继承已认证的凭证，具有与您相同的权限运行。任何给定对象的应用程序许可权与您的许可权是相同的。应用程序产生的进程带有您的标识并将它传递给它们启动的进程。

注: 更改 **setuid** 许可权位仅更改其下运行可执行文件的本地操作系统标识，而不更改网络标识。

一些服务器被编写为作为单独已认证的主体运行。对于这些服务器，系统管理员在注册表数据库中创建一个帐户。在启动这些服务器之后，服务器进程使用注册表认证、接收它的凭证并在它自己的标识，而非您的标识下运行。

ACL 项和掩码

ACL 项具有几种不同的 *ACL 项类型*，每种类型用于一种特定的目的。所有的 ACL 项都以统一的列表语法表示。

ACL 语法

DCE 控制程序使用 Tcl 语法支持的命令语法。在 Tcl 中，表示 ACL 项的列表根据 ACL 项类型包含两种或三种元素，且使用下列格式： x

{*type* [*key*] *permissions*}

图49中的三种样本 ACL 项都使用 Tcl 用于接受输入的格式。

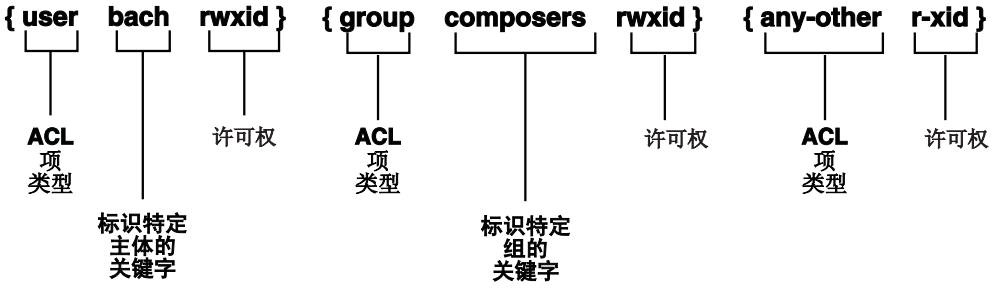


图 49. 样本 ACL 项

第一个样本 ACL 项为本地单元中的主体设置许可权，称为 **bach**。ACL 项类型是 **user**，密钥是 **bach**，许可权是 **rwxid**。项组件由空格字符分隔。

第二个样本 ACL 项为本地单元中的组设置许可权，称为 **composers**。ACL 项类型是 **group**，密钥是 **composers**，许可权是 **rwxid**。

第三个样本 ACL 项为本地单元或外部单元（除非它们与非常特定项匹配）中的所有其它主体设置许可权。ACL 项类型是 **any-other**，没有密钥，许可权是 **r-xid**。不是所有的 ACL 项类型都需要密钥。

在输出上，ACL 许可权的 Tcl 格式包含每个可能的许可权的许可权字符或 - （破折号）。两个示例是:

```
{user mozart crwx---}
{user brahms -------}
```

对于输入，可以接受使用输出格式，或可以使用省略破折号的简易表单格式。对于输入，相同的示例可缩减为:

```
{user mozart crwx}
{user brahms -}
```

单一破折号被保留为显示用户 **brahms** 被所有许可权拒绝。

主体和组的 ACL 项类型

ACL 项类型允许您为下列内容定义项:

- 主体和组
  - 本地单元中的主体和组
  - 外部单元中的主体和组
  - 代理项
  - 其中没有创建个别 ACL 项的本地单元中的所有主体
  - 特权属性与任何其它 ACL 项都不匹配的本地和所有外部单元中的所有主体
- 用于已认证和未认证的用户的掩码
- 不同 DCE 发行版可以复制和显示（如果没有解释）的到目前为止未定义的项类型

如果存在任何没有认证的主体或组，项中的许可权将由**未认证**掩码（在本章以后章节中描述）进一步限制。已认证主体的所有项，除了 **user_obj** 和 **other_obj** 项，将由 **mask_obj** 掩码（也在本章以后章节中描述）进一步限制。

下列列表显示主体和组的项类型、它们的意义及其项格式。所有 ACL 都具有定义于其中的缺省单元，正如在表中所述。它可以更改，用于定义各种数据类型单元。

此列表使用下列语法变量:

principal_name
注册表数据库中主体的名称

group_name
注册表数据库中定义的组名称

cell 使用格式 */.../name* 的单元的全局路径名。

permissions
对象 ACL 管理器可用的许可权。

下列是主体和组 ACL 项类型:

user_obj
为对象的真实或有效用户建立许可权。示例是文件的所有者。项格式是:

{**user_obj** *permissions*}

group_obj
为对象的真实或有效组的成员建立许可权。示例是文件的组。项格式是:

{**group_obj** *permissions*}

other_obj
为缺省单元中所有其它主体建立许可权，除非它们在项类型为 **user** 的 ACL 中特别命名、是具有项类型 **group** 的 ACL 中命名的组成员或与 **user_obj** 或 **group_obj** 项指定的主体匹配。项格式是:

{**other_obj** *permissions*}

user 为 ACL 缺省单元中特定主体建立许可权。此 ACL 项类型要求密钥是主体名称。项格式是:

{**user** *principal_name permissions*}

group 为缺省单元中特定组成员建立许可权。此 ACL 项类型要求密钥是组名。项格式是:

{**group** *group_name permissions*}

foreign_user

为外部单元，不是 ACL 缺省单元中特定的主体建立许可权。必须提供主体名称和单元名称作为密钥来标识主体。项格式是:

{**foreign_user** *cell_name*/*principal_name* \ *permissions*}

foreign_group

为外部单元，不是 ACL 缺省单元中特定的组建立许可权。必须提供组名和单元名称作为密钥来标识组。项格式是:

{**foreign_group** *cell_name*/*group_name permissions*}

foreign_other

为特定外部单元，不是 ACL 缺省单元中的其它主体建立许可权，该 ACL 缺省单元在项类型 **foreign_user** 的 ACL 项中没有特别命名或是 ACL 项类型 **foreign_group** 中命名的组成员。必须提供单元名称作为密钥来标识外部单元。项格式是:

{**foreign_other** *cell_name permissions*}

any_other

为本地单元或外部单元（除非它们与 ACL 中更加特定的项匹配）中的所有其它主体建立许可权。项格式是:

{**any_other** *permissions*}

user_obj_delegate

为对象的真实或有效用户的代理人建立许可权。项格式是:

{**user_obj_delegate** *permissions*}

group_obj_delegate

为对象的真实或有效用户成员的代理人建立许可权。项格式是:

{**group_obj_delegate** *permissions*}

other_obj_delegate

为缺省单元中所有其它主体的代理人建立许可权，除非它们在项类型为 **user** 的 ACL 中特别命名、是具有项类型 **group** 的 ACL 中命名的组成员或与 **user_obj** 或 **group_obj** 项指定的主体匹配。项格式是:

{**other_obj_delegate** *permissions*}

user_delegate

为 ACL 缺省单元中特定主体的代理人建立许可权。此 ACL 项类型要求密钥是主体名称。项格式是:

{**user_delegate** *principal_name permissions*}

group_delegate

为缺省单元中特定组成员的代理人建立许可权。此 ACL 项类型要求密钥是组名。项格式是:

{**group_delegate** *group_name permissions*}

foreign_user_delegate

为外部单元，不是 ACL 缺省单元中特定主体的代理人建立许可权。必须提供主体名称和单元名称作为密钥来标识主体。项格式是:

{**foreign_user_delegate** *cell_name/principal_name* \ *permissions*}

foreign_group_delegate

为外部单元，不是 ACL 缺省单元中特定组的代理人建立许可权。必须提供组名和单元名称作为密钥来标识组。项格式是:

{**foreign_group_delegate** *cell_name/group_name* \ *permissions*}

foreign_other_delegate

为特定外部单元，不是 ACL 缺省单元中的其它主体的代理人建立许可权，该 ACL 缺省单元在项类型 **foreign_user** 的 ACL 项中没有特别命名或是 ACL 项类型 **foreign_group** 中命名的组成员。必须提供单元名称作为密钥来标识外部单元。项格式是:

{**foreign_other_delegate** *cell_name permissions*}

any_other_delegate

为本地单元或外部单元（除非它们与 ACL 中更加特定的项匹配）中的所有其它主体的代理人建立许可权。项格式是:

{**any_other_delegate** *permissions*}

组许可权和项目列表

项目列表的主体增加组许可权，该列表是主体或别名为其成员的所有组的列表。当主体尝试访问对象时，主体具有被授权是 ACL 中某一项的所有组的各种许可权的逻辑“或”所形成的访问权，且在这些组中主体还是其中一个成员。注意，主体仅从主体登录使用的名称或别名，而不是同时从名称和别名收取访问权利。(有关别名和项目列表的详细信息，请参阅第283页的『第30章 创建和维护主体、组和组织』。)

例如，假设 ACL 包含下列项:

```
{user_obj crwxid-}
{group_obj crwx---}
{other_obj -r-----}
{group composers crwx---}
{user bach crwx---}
{user mozart crwx---}
{group performers --w-idt}
```

用户 **cole** 是组 **composers** 和组 **performers** 的成员。因为 **cole** 从这两个组收取许可权，因此他的许可权为 **crwxidt**。(安全性服务提供一种方法，防止组被包含入项目列表中，这样来防止组许可权作为项目列表的一部分被收入。详细信息请参阅第283页的『第30章 创建和维护主体、组和组织』。。)

使用主体和组 ACL 项

当安全性机制应用 ACL 时，将以特定次序选择 ACL 项。最特定项在较次级特定项之前选择。

在为主体和组使用 ACL 项类型时，考虑 **user_obj**、**group_obj** 和 **other_obj** 类型与 **user**、**group** 和 **other** 的 POSIX 文件许可权相似。使用 **user** 和 **group** 类型为特定主体或组指定许可权。

user_obj、**group_obj**、 **other_objuser** 和 **group** 项类型适用于 ACL 缺省单元中的主体和组。要为外部单元中的特定主体和组设置许可权，请使用 **foreign_user** 和 **foreign_group** 项。这些项在外部单元中使用与 **user** 和 **group** 项在缺省单元中所用

的相同方法来设置许可权。使用 **foreign_other** 为外部单元中其它项设置许可权，使用的方法与缺省单元中 **other_obj** 为其它项设置的方法相同。

any_other 项类型为其它项类型不适用的所有本地和外部主体设置许可权。如果已显式或隐式地为本地或外部主体设置了任何其它项类型，则 **any_other** 项将不适用。这是因为当管理器在主体和项之间找到了匹配，则它将停止检查 ACL 列表，及应用于找到的项（或找到的组项）。除了下面描述的掩码类型，所有其它 ACL 项类型都由 ACL 管理器检查以查看在 ACL 管理器检查 **any_other** 项类型之前是否存在匹配。有关 ACL 检查次序的详细信息，请参阅第269页的『ACL 项的检查序列』。。

ACL 掩码的项类型

ACL 项中的掩码建立能授予主体的最大许可权。有两种掩码： **mask_obj 掩码**和**未认证掩码**。仅授予 ACL 项中给出的许可权和掩码。例如，如果 ACL 项指定 **rwx** 许可权且掩码仅指定 **x** 许可权，则许可权将和掩码进行“与”操作，且仅授予 **x** 许可权。

如果存在 **mask_obj** 掩码，则应用除 **user_obj** 和 **other_obj** 以外的所有项类型。**未认证**掩码将应用到所有未认证主体。因为 ACL 管理器是从 ACL 项派生许可权的，因此它通过 **mask_obj** 掩码（如果存在的话）过滤出每种许可权，最后还要通过**未认证**掩码去过滤。管理器仅授予第一个匹配项、**mask_obj** 掩码和**未认证**掩码中的那些许可权。

注: 如果不创建**未认证**掩码，则未认证主体将拒绝对对象的所有访问。如果因为用户没有 DCE 凭证而未认证，则用户匹配的唯一项为 **any_other** 项类型，该类型随后由**未认证**掩码屏蔽。这意味着为使这种未认证用户对对象具有访问权，对象的 ACL 必须包含 **any_other** 类型项和**未认证**掩码项。

下面是掩码用法的示例。对于特定对象，存在大量指定对该对象进行 **rw** 访问的 ACL 项。需要临时限制只读访问，但不必更改所有 ACL 项。只要简单地创建 **r** 的 **mask_obj** 掩码，然后在完成后删除，提供临时限制。

不同 DCE 发行版的 ACL 项类型

extended 项类型为 ACL 项提供类属格式，该 ACL 项允许以后 DCE 发行版实现新的 ACL 项类型。因为新的类型包含在 **extended** 项的类属格式中，所以即使早期 DCE 发行版不能解释新的项类型的意思，也能复制、显示和打印它们。

第272页的『复制 ACL』 说明如何复制扩展项。注意不能修改扩展项；但是可以删除它们。

extended ACL 项具有下列格式:

{**extended** *uuid.ndr.ndr.ndr.ndr.number_of_bytes.data*
permissions}

其中:

uuid 标识扩展 ACL 项的项类型的 UUID。（此 UUID 可以标识本文档中描述的一种 ACL 项类型或到目前为止未定义的 ACL 项类型。）

ndr.ndr.ndr.ndr
标识数据编码的网络数据表示法 (NDR) 格式标签（以十六进制格式，用点分隔）。

number_of_bytes

指定 *data* 中总字节数的十进制数。后面跟随一个点。

data 十六进制格式的 ACL 数据。（ACL 数据的每个字节都是两个十六进制位。）ACL 数据包含除了许可权以外的所有 ACL 项规范说明。ACL 数据没有解释；它假设数据传送至的 ACL 管理器能理解该数据。

permissions

将由项授予的许可权。

ACL 项的检查序列

ACL 管理器读取 ACL 项列表以查找适用于尝试执行特定操作的某个个人的特定项。ACL 管理器首先查找主体的特权属性之间的匹配，或处理期望的访问和 ACL 中列出的特权属性。当管理器找到一次匹配时，如果 **mask_obj** 掩码存在的话，它将在匹配 ACL 项中检查许可权，将 **mask_obj** 掩码应用于它（除非它是类型 **user_obj** 或 **other_obj** 的项。最后，如果主体未经认证的话，ACL 管理器将应用**未认证**掩码（如果存在的话）。如果产生的许可权允许所请求的访问，管理器会将它授予主体。否则，将拒绝访问。

因为当 ACL 管理器找到一次匹配时，它将停止检查 ACL 项，所以知道检查 ACL 的次序很重要。第270页的图50 显示检查次序和应用的掩码。ACL 管理器以下列次序检查项，除非启动程序主体不对 **..._delegate** 项进行检查。代理主体对所有项都进行检查。

1. 首先，ACL 管理器以下列次序检查用户 ACL 项:
   - **user_obj**
   - **user_obj_delegate**
   - **user**
   - **user_delegate**
   - **foreign_user**
   - **foreign_user_delegate**

   ACL 管理器停止在找到的第一个匹配用户项中进行的所有项检查，将许可权应用到项中。用户项以以前列表中所显示的次序检查，从最特定的到最不特定的。
2. 如果 ACL 管理器在用户项中没有找到匹配，它将检查下列*所有*组项:
   - **group_obj**
   - **group_obj_delegate**
   - **group**
   - **group_delegate**
   - **foreign_group**
   - **foreign_group_delegate**

   如果任何组 ACL 项与主体的项目列表都匹配，且这些项的许可权逻辑 “OR” 授权访问，则授权访问且不执行进一步检查。

   因为委托人从列在他们是其中一个成员的 ACL 中的所有组中收取许可权（而且他们还在项目列表中），因此将检查*所有的*组并逻辑 “OR” *所有的*主体组许可权。组项检查次序无关紧要。有关项目列表的详细信息，请参阅第267页的『组许可权和项目列表』。
3. 如果 ACL 管理器没有在主体请求许可权和组项中的组成员之间找到匹配，它将检查 **other_obj** 和 **other_obj_delegate** 项。如果 ACL 管理器找到了匹配，它将停止检查 ACL 项。

4. 如果 ACL 管理器没有在主体请求许可权和 **other_obj** 或 **other_obj_delegate** 项之间找到匹配，它将检查 **foreign_other** 和 **foreign_other_delegate** 项。如果 ACL 管理器找到了匹配，它将停止检查 ACL 项。
5. 如果 ACL 管理器没有在主体请求许可权和 **foreign_other** 或 **foreign_other_delegate** 项之间找到匹配，它将检查 **any_other** 和 **any_other_delegate** 项。如果它在 **any_other** 或 **any_other_delegate** 项中没有找到匹配，它将拒绝所有对对象的访问。

最后许可权是启动程序主体许可权和每个代理许可权的逻辑乘。

图50 显示它们应用到 ACL 项的这些步骤。其中两列区分了没有被 **mask_obj** 屏蔽和被其屏蔽的 ACL 项。

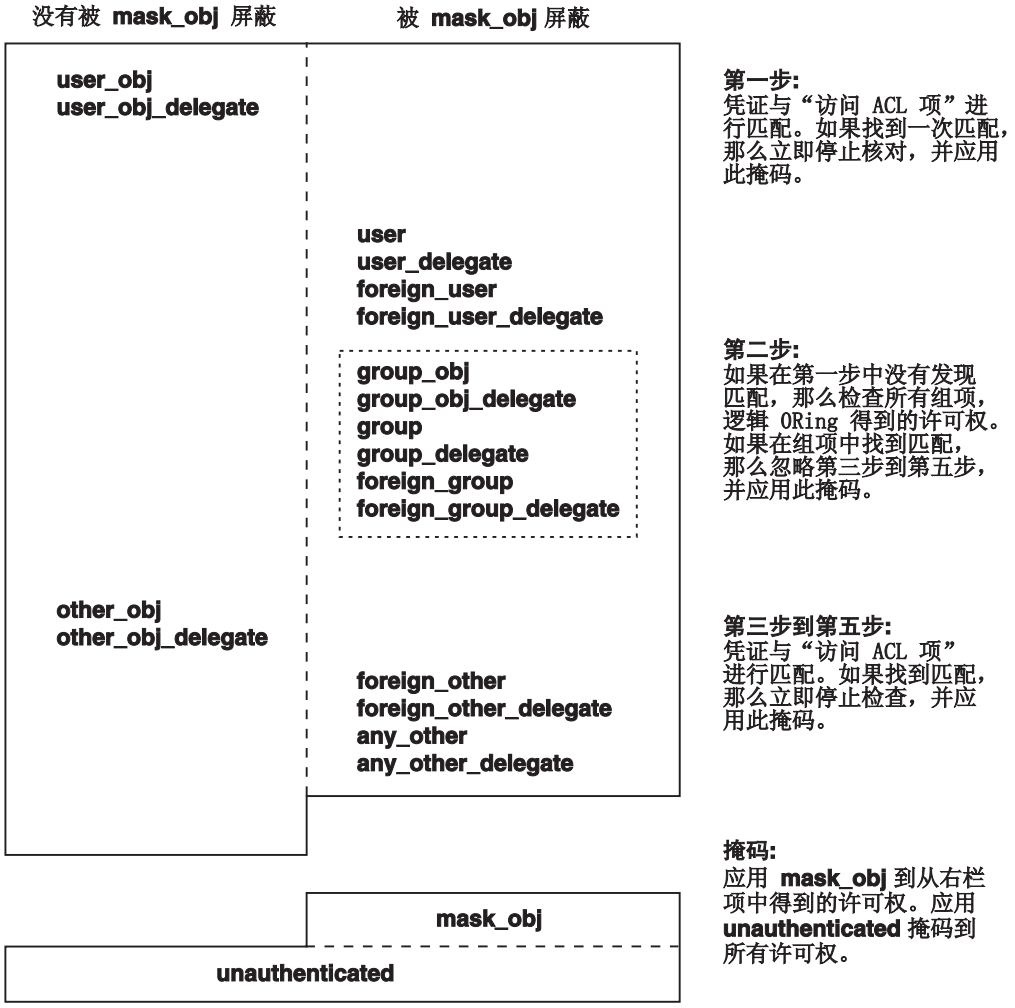


图 50. 检查 ACL 和应用掩码的次序

mask_obj 掩码和 ACL 检查

在 ACL 管理器授予任何由检查 ACL 项导出的许可权之前，它通过 **mask_obj** 掩码过滤象许可权。仅在 ACL 项和掩码中命名的那些许可权才可授出。例如，如果 ACL 项授出 **rwx** 许可权且 **mask_obj** 项仅指定 **r** 和 **w** 许可权，则仅 **r** 和 **w** 可授出。忽略 ACL 项中命名为 **x** 的许可权。

未认证掩码和 ACL 检查

如果 ACL 管理器从未认证主体接收访问请求，它将检查 ACL 项并按前面所述应用 **mask_obj** 掩码（如果可用的话）。然后通过未认证主体（**未认证**项类型）的掩码过滤结果产生许可权。仅授权**未认证**掩码、ACL 项和 **mask_obj** 掩码（如果存在的话）中指定的那些许可权。

授出许可权时检查次序的影响

可以考虑 ACL 项检查次序为从最特定到最不特定。例如，假设 ACL 包含下列项:

```
{user mahler r}
{group composers rwx}
```

如果命名为 **mahler** 的主体（它是组 **composers** 的成员）请求执行 (**x**) 访问，它将被拒绝。发生这种情况是因为检查次序指定在检查所有组 (**group_obj**、**group** 和 **foreign_group**) 项之前，先检查所有用户项 (**user_obj**、**user** 和 **foreign_user**)。因此，ACL 管理器找到的第一个匹配是用户 **mahler** 和用户 **mahler** 的 ACL 项之间的匹配。当找到匹配用户项后，检查停止并授出找到的许可权。在这种情况中，检查在具有更宽松的许可权的 **group** 项之前停止。

拒绝访问

当为主体或组创建 ACL 项时，您仅授予在 ACL 项中指定的许可权。要拒绝对对象具有所有访问权的主体，请创建包含破折号代替许可权的 ACL 项。例如，要拒绝对用户 **mozart** 的所有访问，项将为:

```
{user mozart -}
```

如果选择拒绝对特定主体或组的访问，请选择可用的最特定项类型。通常对主体来说，这是 **user** 或 **foreign_user** 的项类型；对组来说，是 **group** 或 **foreign_group** 的项类型。注意，如果主体是对象的所有者或对象组的成员，则必须使用 **user_obj** 或 **group_obj** 项类型来保证拒绝了访问。

要拒绝对所有未认证用户的访问，请不要创建**未认证**掩码。如果未创建此掩码（**未认证**的 ACL 项类型），则仅已认证主体能访问对象。通过创建没有许可权（或用破折号代替许可权）的**未认证**掩码来完成相同的行为。此方法还具有使用图例说明未认证用户不具有访问权的附加优点。

ACL 管理任务

ACL 管理包含为 DCE 实体上的 ACL 创建、修改和删除项。可以使用 DCE 控制程序执行所有这些任务。控制程序的 **acl** 命令在 ACL 上执行这些下列操作:

- 在本地单元和外部单元中创建和修改 DCE 对象的 ACL 项。（注意，当创建对象时，它们与初始 ACL 项关联。详细信息请参阅第272页的『从文件生成 ACL』。）
- 显示对象 ACL 管理器为对象实施的许可权。
- 创建和修改用于限制可允许的许可权的掩码。

注: 显示和处理 UNIX 方法的标准 UNIX 工具仅在为文件系统建立的 ACL 上有效果。

有关 DCE 控制程序的 **acl** 命令的详细描述，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference*。

复制 ACL

要将 ACL 从一个 DCE 对象复制到另一个，请使用具有 **-acl** 选项的 DCE 控制程序 **acl replace** 命令，如下显示:

```
dcecp> acl replace  /.:/hosts/hermes  -acl [acl show /.:/hosts/cyclops]
dcecp>
```

示例命令将主机 **cyclops**（它的名称在 **-acl** 选项调用的 **acl show** 命令中指定）的 ACL 替换为主机 **hermes** 的 ACL。注意 **-acl** 选项中的 **-acl show** 命令是如何被附上 [ ]（方括号）的。当 **-acl** 选项值是命令调用时需要它。

如果正在单元之间复制，则使用 **acl replace** 命令的 **-cell** 选项和它的 **-acl** 选项。例如:

```
dcecp> acl replace /.:/hosts/hermes -acl [acl show /.:/hosts/cyclops]
\
> -cell [acl show /.:/hosts/cyclops -cell]
dcecp>
```

要将 **extended** 项类型从一个 ACL 管理器域复制到另一个 ACL 管理器域，请将 **dcecp acl show** 命令的输出作为至 **acl replace** 命令的输入。要以这种方式复制 **extended** 项，两个 ACL 管理器必须支持 **extended** 项类型。

从文件生成 ACL

创建 ACL 的简便方法是创建和编辑文本文件，以便它包含所期望的 ACL 项，然后通过使用 **acl replace** 命令从它生成 ACL。

例如，假设文件 **std_acl** 包含下列项:

```
mask_obj:crwxid-
user_obj:crwxid-
group_obj:crwx---
other_obj:-r-----
user:lizt:crwx---
group:composers:-r-----
user:bach:crwx---
user:mozart:crwx---
```

下列 **acl replace** 命令将 **std_acl** 中的项添加到名称为 **/.../dresden.com/my_filesystem/opus** 的 ACL:

```
dcecp> acl replace /.../dresden.com/my_filesystem/opus -acl [cat
std_acl]
dcecp>
```

acl replace 命令使用来自文件 **std_acl** 的 ACL 项覆盖所有的 ACL 项。无论它们以前是什么样，**opus** 的 ACL 现在为:

```
mask_obj:crwxid-
user_obj:crwxid-
user:lizt:crwx---
user:bach:crwx---
```

```
user:mozart:crwx---
group_obj:crwx---
group:composers:-r-----
other_obj:-r-----
```

容器 ACL

注: 有关 “DCE 安全性注册表” 和 “LDAP 集成” 功能的主题信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

对象 ACL 控制对对象本身的访问。除了它的对象 ACL 以外，容器对象还具有初始容器 ACL 和初始对象 ACL。这两个 ACL 不是用于访问控制，而是用于复制对象的初始 ACL 或初始容器内创建的容器。通过将 **-ic** 和 **-io** 选项用于 **dcecp acl** 命令，可以使用与平常 ACL 相同的方式编辑初始容器 ACL 和初始对象 ACL。

对象和容器

用于对象的 ACL 类型取决于对象是简单对象还是容器。容器是保持对象的对象。保持的对象可以为简单对象或容器对象。简单对象不保持其它对象。虽然任何 DCE 组件都具有对象和容器，最简单和最常用的举例说明是文件系统。在文件系统中，这些是文件和目录。文件是简单对象，目录是容器。目录可以保持简单对象（文件）和其它容器（子目录）。

对象 ACL 与简单对象和容器对象关联。初始容器和初始对象 ACL 仅与容器对象关联。

对象和容器的初始 ACL

初始 ACL 项和包含它们的 ACL 在创建对象时将自动应用。可以使用 DCE 控制程序在任何时间修改项。用作容器和对象的初始 ACL 的 DCE ACL 类型如下所示:

- 初始容器 ACL 确定在容器内创建的容器的缺省 ACL。例如，目录的文件系统初始容器 ACL 指定该目录中创建的子目录的缺省 ACL。
- 初始对象 ACL 确定在容器内创建的对象的缺省值。例如，目录的文件系统初始对象 ACL 指定该目录中创建的文件的缺省 ACL。

对象的缺省 ACL

当在容器中创建简单对象时，它将容器的初始对象 ACL 继承为它的对象 ACL。 第274页的图51 说明缺省 ACL 如何赋值给容器中创建的简单对象。

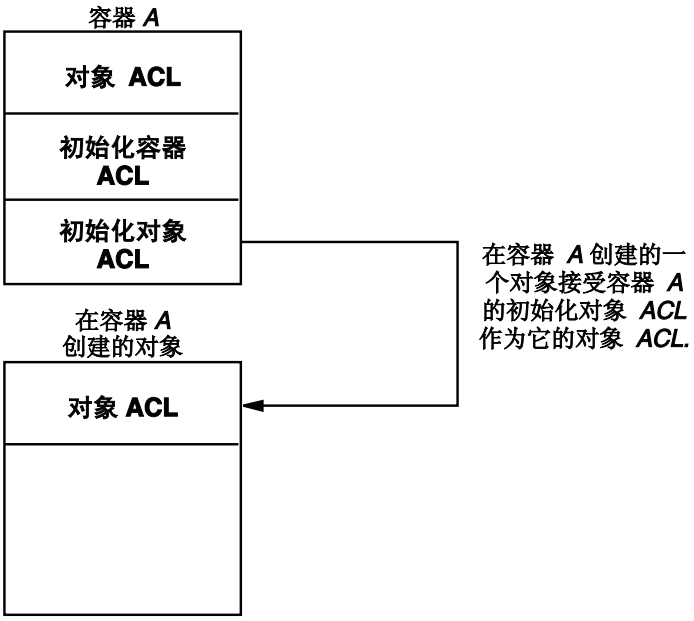


图 51. 容器中创建的对象的初始 ACL

容器的缺省 ACL

当在容器（例如目录中的子目录）内创建容器时，它将继承父代容器特性:

- 作为其对象 ACL 和其“初始容器 ACL”的初始容器 ACL
- 作为其初始对象 ACL 的初始对象 ACL

例如，如果在目录 **marketing** 中创建名称为 **report** 的文件，系统将 **report** 赋值为目录 **marketing** 的初始对象 ACL。如果在 **marketing** 中创建子目录，系统将指定新子目录为 **marketing** 的初始容器 ACL。新子目录同时接收匹配父目录的初始 ACL 的一组初始 ACL。在这个示例中，新的子目录还将 **marketing** 的初始 ACL 接收为它自己的 ACL。第275页的图52 说明缺省 ACL 如何赋值给容器中创建的对象。

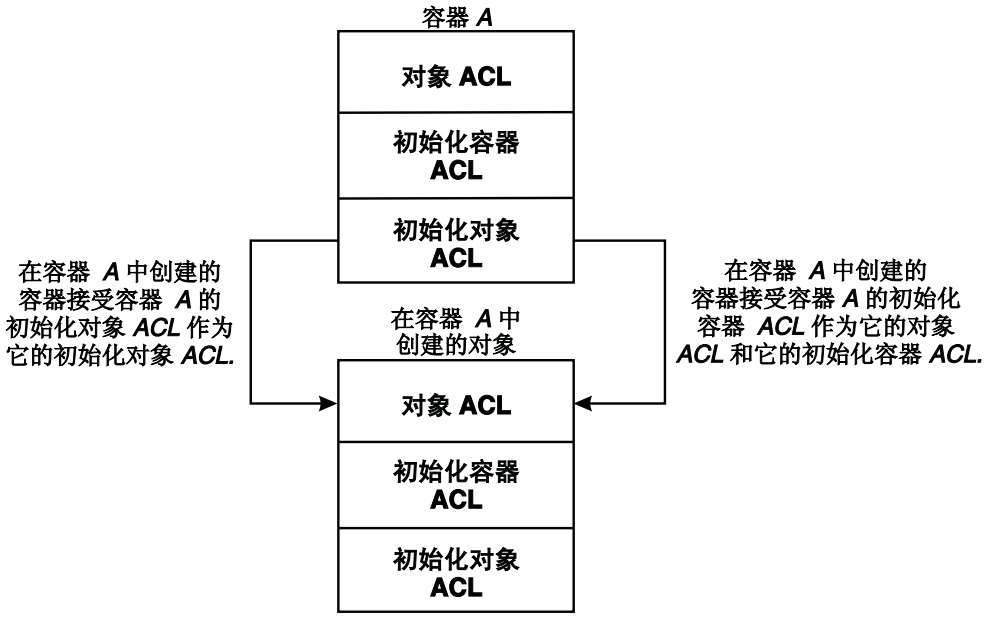


图 52. 容器中创建的容器的初始 ACL

缺省容器 ACL 示例

下列示例显示 ACL 如何初始地赋值给容器内创建的容器。

假设容器 A 具有下列 ACL:

对象 ACL

```
{user_obj crwxid}
{group_obj crwxid}
{other_obj r}
```

初始容器 ACL

```
{user_obj crwxid}
{group_obj rw}
{other_obj r}
```

初始对象 ACL

```
{user_obj crwxid}
{group_obj r}
{other_obj r}
```

当容器 B 在容器 A 中创建时，它具有下列缺省 ACL:

对象 ACL（容器 A 的初始容器 ACL）

```
{user_obj crwxid}
{group_obj rw}
{other_obj r}
```

初始容器 ACL（容器 A 的初始容器 ACL）

```
{user_obj crwxid}
{group_obj rw}
{other_obj r}
```

初始对象 ACL（容器 A 的初始对象 ACL）

```
{user_obj crwxid}
{group_obj r}
{other_obj r}
```

编辑 ACL 时掩码的效果

如果用户指定新的 **mask_obj** ACL 项，则 **acl modify** 将使用它。否则，**acl modify** 命令将使用下列段落中显示的算法来重新计算掩码，除非用户已指定了选项 **-mask** calc、 **-mask** nocalc 或 **-purge** 之一。因此，掩码可以在每个 **acl modify** 命令上更改、授予更多或更少的许可权。

这里是当计算掩码时，**acl modify** 命令使用的算法。

1. 检索文件的现有 ACL。
2. 执行所有请求以删除项和减少现有项的许可权。
3. 计算所有剩余项的实际许可权联合。
4. 确定在实际的和有效的权限之间有哪些不同的许可权。（这是步骤 3 和 4 结果的逻辑 XOR。）
5. 执行所有请求，将新项添加到 ACL 和所有请求以增加现有项的许可权。
6. 计算这些新授予的许可权和旧的有效许可权的联合（从步骤 4）。这是候选的新掩码值。
7. 如果在候选新掩码中存在任何许可权，同时在原始实际权限和有效权限之间也有这些不同（从步骤 5）的许可权中存在，则应用新掩码将意外赋予用户不想赋予的一些新权利。除非用户指定选项 **-mask calc**、**-mask nocalc** 或 **-purge** 之一，否则此条件将为一个错误，且不修改 ACL。否则，候选新掩码将作为新掩码应用。

对于大多数 ACL 操作，这种自动重新计算是安全的。在某些罕见的情况中，重新计算掩码将赋予用户不想要的附加权利；例如，将用户未指定的许可权授予项，且这些许可权又不在项的以前有效的权限中。

下列示例显示以什么方式重新计算掩码以及选项的效果。

注意，ACL 包含授予 **rwx** 许可权给某些用户的项，但掩码允许 **r-x** 有效许可权。对 **rwx**，添加新的 **rwx** ACL 项并重新计算掩码（根据步骤 6）是不安全的，因为第一个用户的有效访问权意外地从 **r-x** 更改为 **rwx**。如果 **acl modify** 命令检测到这种不安全条件，它的缺省操作是发出错误消息，不更改 ACL。

显示许可权和有效许可权的初始状态为:

```
dcecp> acl show /.:/concertos
{user vivaldi rwx effective r-x}
{mask_obj r-x}
dcecp>
```

如下所示添加一个用户将导致错误是因为掩码重新计算将给予 **vivaldi** 有效的 **rwx** 许可权:

```
dcecp> acl modify /.:/concertos -add {user telemann rwx}
Error: Unintended permissions not granted.
dcecp>
```

尽管新的许可权授予 **vivaldi**，**-mask calc** 选项的显式使用仍允许应用重新计算的掩码。掩码设置为授予 ACL 上文件组类项许可权的联合。此选项可能导致无意间授予附加的许可权。

```
dcecp> acl modify /.:/concertos -add {user telemann
rwx} -mask calc
dcecp> acl show /.:/concertos
{user vivaldi rwx effective rwx}
{user telemann rwx effective rwx}
{mask_obj rwx}
dcecp>
```

显式使用 **-mask nocalc** 选项保留 **r-x** 掩码，导致减少 **telemann** 的有效许可权。如完全按用户指定修改 ACL，就不会发生掩码计算或清除许可权。

```
dcecp> acl modify /.:/concertos -add {user telemann rwx} -mask nocalc
dcecp> acl show /.:/concertos
{user vivaldi rwx effective r-x}
{user telemann rwx effective r-x}
{mask_obj r-x}
dcecp>
```

使用 **-purge** 选项替换所有项中具有有效许可权的实际许可权。更精确地说，如果命令检测出一个不安全条件，条件将所有现有掩码的当前掩码值和文件组类中未修改的项相“交”，将所有 ACL 项（除了 **user_obj**、**other_obj**、 **mask_obj** 和 **unauthenticated**）替换为它们的有效许可权。

```
dcecp> acl modify /.:/concertos -add {user telemann rwx} -purge
dcecp> acl show /.:/concertos
{user vivaldi rwx effective r-x}
{user telemann rwx effective rwx}
{mask_obj rwx}
dcecp>
```

第29章 管理 DCE 安全性服务的控制程序

可以通过使用 DCE 控制程序 (**dcecp**) 执行 DCE 安全性服务的大多数管理任务。但是，此服务的一些组件需要您使用 DCE 提供的其它控制程序。

本章提供 DCE 控制程序为 DCE 安全性服务管理提供的命令信息。本章还描述了注册表编辑器程序 (**rgy_edit**) 为维护本地注册表提供的命令。

经常用于与安全性有关的管理任务的控制程序，例如 **password_export** 和 **sec_create_db** 在本章中不适用。这些程序在本指南随后的章节中与执行任务的指令一起描述。

使用 DCE 控制程序

因为有关 DCE 控制程序和命令语法的详细信息出现在本指南的第1页的『第1部分 DCE 控制程序』 中，所以本章不重复这些信息。它仅提供 DCE 控制程序为具体管理 DCE 安全性服务所提供的命令。

DCE 控制程序为 DCE 安全性服务的网络范围注册表（注册服务组件）创建和维护主体、组、组织和帐户。控制程序还在保护本地节点上安全性服务器口令的键表文件上操作（认证服务组件）。另外，它还维护保护 DCE 资源的 ACL（特权服务组件）。管理 DCE 安全性服务的 DCE 控制程序命令通过它定义的各种对象在这些安全性和 DCE 范围资源上操作。例如，控制程序 **acl check** 命令显示 DCE 安全性服务对象赋予调用主体的许可权。

下列章节描述 DCE 控制程序操作的 DCE 安全性服务对象以及控制程序可以在这些对象上执行的操作类型。

安全性服务对象

DCE 控制程序具有在下列安全性服务组件上操作的功能:

principal
此对象表示注册表主体。这些主体可以是网络上的用户、网络上的服务器、网络上的机器或本地单元将与之一起从事交叉单元认证的单元。

group 此对象表示注册表组。组是可认为其将访问权赋予对象的主体集合。

organization
此对象表示注册表组织。组织是可向其分配策略，扩充管理控制区域的主体集合。

account
此对象表示为主体在注册表中建立的帐户。

registry
此对象表示 DCE 单元中的注册表或客户信息的 DCE 安全性服务数据库。所操作的注册表副本可以是主副本或从属副本。

xattrschema
对于为本地主机上主机守护程序 (**dced**) 维护的 DCE 安全性服务组件和数据指定的扩展注册表属性 (ERA)，此对象在模式或定义上操作。

acl 此对象表示可以受 DCE 安全性服务的 ACL 设施保护的所有 DCE 实体的 ACL。

keytab
此对象表示为 DCE 安全性服务中已认证的服务器主体存储密钥或口令的文件。

DCE 安全性服务的 DCE 控制程序操作

表16 列出 **dcecp** 在 DCE 安全性服务对象上执行的操作。

使用 DCE 控制程序命令创建和维护本指南的第283页的『第30章 创建和维护主体、组和组织』 和 第311页的『第31章 创建和维护帐户』 中给出的主体、组、组织和帐户。

表 16. DCE 安全性服务的 DCE 控制程序操作

操作	描述
add	将主体添加到组或将组织添加到注册表副本。
catalog	显示注册表副本所有主体、组和组织的名称。对于注册表本身，显示 DCE 单元中现有的主副本和从属副本。
check	显示 DCE ACL 当前赋予安全性主体的许可权。
checkpoint	复位注册表检入间隔。
create	在注册表副本中创建新的主体、组、组织或帐户。同时为 ERA 模式创建新的项。
delete	从注册表副本中删除主体、组、组织或帐户。对于注册表本身，删除从属副本。对于 ERA 模式，删除项。对于 DCE ACL，删除 ACL 项。
designate	注册表副本为主副本的更改。
destroy	破坏指定的副本及其注册表数据库的副本。
disable	禁用用于更新的注册表主副本。
dump	显示单元中现有的每个注册表副本的信息。
enable	启用用于更新的注册表主副本。
generate	为现有的注册表帐户生成随机口令。
help	显示有关主体、组、组织、帐户、ERA 模式或注册表副本中的 DCE ACL，以及有关注册表副本本身的帮助信息。
list	显示注册表副本中从属于组或组织的主体名称。
modify	在注册表副本中为主体、组、帐户、ERA 模式项、DCE ACL 项或注册表本身修改属性信息。同时还对组织修改策略信息。
operations	显示可由或可在主体、组、组织、帐户、ERA 模式、DCE ACL 或注册表副本执行的操作。
permissions	显示由受保护 DCE 组件上 ACL 授予的许可权。
remove	从注册表副本中的组或组织删除一个或多个主体。
rename	在注册表副本中更改的主体、组、组织或 ERA 模式的名称。
replace	替换 DCE 组件上的整个 ACL 或注册表副本的地址。
show	显示有关主体、组、ERA 模式项或 DCE ACL 项属性的信息。同时显示有关组织、帐户或注册表副本策略的信息。
stop	停止安全性服务器进程。
synchronize	指示从属注册表副本从主副本更新它的内容。
verify	检查是否所有的注册表副本都是最新的。

使用注册表编辑器

虽然可以使用 DCE 控制程序来维护注册表，但仍能使用注册表编辑器来维护位于单元中主机上的本地注册表。

下列章节说明如何启动、停止注册表编辑器或获取注册表编辑器帮助信息，描述了用于本地注册表维护的命令。使用注册表编辑器维护本地注册表的特定指令在第283页的『第30章 创建和维护主体、组和组织』和第311页的『第31章 创建和维护帐户』中给出。

有关所有注册表编辑器命令的详细描述，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference*。

启动、停止和获取帮助

注: 有关 “DCE 安全性注册表” 和 “LDAP 集成” 功能的主题信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

注册表编辑器以两种方式运行: 交互式和命令行。在交互方式中，控制程序提示您它所需要的信息。在命令行方式中，由您输入控制程序在命令行上需要的所有信息。在命令行方式中，在某一时间只能执行一个操作；但是，您会发现命令行方式在创建执行注册表编辑器命令序列的外壳脚本时非常有用。本指南中的大多数示例都是交互方式的。(有关如何以命令行方式调用和使用注册表编辑器的详细信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference*。)

要以交互方式启动注册表编辑器，请输入下列命令:

```
$ dceshared/bin/rgy_edit
```

注: 如果没有认证过，则 **rgy_edit** 命令将警告您绑定未认证--无法对注册表建立认证。

注册表编辑器通过显示当前注册表站点和 **rgy_edit=>** 提示的名称来响应，如下所示:

```
Current site is:
registry server at /.../bayre.com/subsys/dce/sec/oddball
rgy_edit=>
```

如果名称服务无法提供名称，则缩短输出。例如，替代下列输出:

```
registry server at /.../bayre.com/subsys/dce/sec/oddball
```

显示将为:

```
registry server at /.../bayre.com
```

要从 **rgy_edit** 命令退出，则在命令提示处按 **Return**。例如，要从 **add** 命令退出以添加，则在 **Add Principal=> Enter name:** 提示处按 **Return**。

要从注册表编辑器退出，则在 **rgy_edit** 提示处输入 **q[uit]** 命令:

```
rgy_edit=> q
$
```

rgy_edit help 命令显示帮助信息。如果输入 **help** 或 **h**，注册表编辑器将显示所有命令以及可用主题的列表。例如:

```
rgy_edit=> help
```

本地注册表维护的 rgy_edit 命令

要在本地注册表上查看或执行任何维护任务，必须首先访问它。要访问本地注册表，请使用 **-l** 选项调用注册表编辑器:

```
$ rgy_edit -l
rgy_edit=>
```

在 **rgy_edit** 提示处，输入想执行的特定操作的命令名称。表17列出可用的命令。

表 17. 维护本地注册表的 rgy_edit 命令

命令	功能
del[**ete**]	删除主体、组或组织的项。
prop[**erties**]	显示或更改本地注册表特性。
pu[**rge**]	删除主体、组或组织的失效项。
v[**iew**]	显示主体、组或组织的项。

有关所有注册表编辑器命令的详细描述，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference*。

第30章 创建和维护主体、组和组织

本章说明如何使用 **dcecp** 创建并维护主体、组和组织。 首先将讨论指派到主体、组和组织的名称以及“DCE 安全性服务”内部用来标识注册表对象的“通用唯一标识”(UUID) 的名称。

主体、组、和组织名称

必须命名注册表中的每个主体、组和组织。虽然主体、组和组织可能同名，但是不会有两个主体、两个组或两个组织同名。例如，不能将两个主体命名为 **smith**，但是可以将一个主体命名为 **smith**，将一个组命名为 **smith** 同时再将一个组织命名为 **smith**。

可以最多指派 3 种类型的名称: 主名称、全名和别名。

主名称

注: 有关“DCE 安全性注册表”和“LDAP 集成”特性的主题信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

将主名称指派到主体、组和组织。注册表对象主名称是大部分系统实用程序所使用的可供人们读懂的名称。将主体、组或组织添加到注册表数据库时必须提供主名称。主名称是作为 **principal show** 命令的输入用来查询注册表数据库的关键字段。

全名

可任选地将全名指派到主体、组和组织。使用对象的全名是出于提供信息目的。它典型地描述或扩展主名称以方便用户识别。例如，一个主体可能带有主名称 **jsbach** 和全名 **Johann S.Bach**。一个组织可能带有主名称 **moco** 和全名 **Motet Composers**。

全名只是数据字段。不能使用它查询注册表数据库。创建主体、组或组织本身时可以创建主体、组和组织的全名。

别名

注: 有关“DCE 安全性注册表”和“LDAP 集成”特性的主题信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

别名是主名的可选备用名称。可将别名指派到主体、组但不能指派到组织。作为主名备用名的别名共享同一个 UUID 和 UNIX 标识。（UUID 和 UNIX 标识在第284页的『通用唯一标识和 UNIX 标识』中描述。） 别名是可用来查询注册表数据库的关键字段。

因为可以为每个主名和每个别名创建帐户，所以别名使您能灵活地为同一主体建立几个帐户。例如，为主名 **mahler** 创建了 3 个别名: **gustav**、**gus** 和 **gm**。然后可为该主体**mahler** 创建 4 个帐户: 为主名创建一个，为每个别名各创建一个。这些帐户可使用不同的主目录和口令并能与不同的组和组织关联。

因为主体只生成与用来登录的主名或别名相关联的权限，这些同一个人的多个帐户能容纳不同的访问模式。例如， **mahler** 可能是 **composers** 组的成员，**gustav** 可能是

music_admin 组（系统管理员组）的成员。主体 **mahler** 作为 **mahler** 登录执行日常任务，还可作为 **gustav** 执行管理职责。要有助于防止系统的意外损坏，好的方法是设置帐户，这样用户可登录到带有执行任务所需最小特权的帐户。

对于组，如果想关联两个带有相同的 UNIX 号的组则别名将非常有用。

参阅第309页的『创建和维护主体或组的别名』。for information on creating aliases.

名称格式

注册表中的名称可能包含任何字符或数字，除了 @ （at 标记）和 **:**（冒号）字符。在名称中避免使用空格、{}（花括号）或 " "（引号），**dcecp** 可能无法正确分析该名称。

名称中允许的字符最大数为 1024。

保留的主体和帐户

保留一些主体和帐户供各种系统操作使用。无法删除保留的主体。可以修改但不能直接删除保留的帐户。但是注意，可以通过删除帐户中指定的组或组织来间接删除保留的帐户。（参阅第311页的『第31章 创建和维护帐户』以获取详细信息。）

保留的主体和帐户的列表如下。在列表中， *cell_name* 是您的单元名称。

- 保留的主体:
  - **dce-ptgt**
  - **krbtgt/***cell_name*
  - **dce-rgy**
- 保留的帐户:
  - **dce-ptgt none none**
  - **krbtgt/***cell_name* **none none**
  - **dce-rgy none none**

对象创建限额

可向每个主体指派对象创建限额。该指派允许您控制主体能够创建的注册表对象数。如果允许用户创建它们自己的组，例如可使用该限额限制它们可以创建的组总数。缺省的对象创建限额是 **unlimited**，意味着对于主体能够创建的对象数没有限制。0（零）值将禁止主体创建任何注册表对象。

每次主体创建注册表对象时主体的对象创建限额减 1。对象创建限额到 0 时将禁止主体创建注册表对象除非使用 **dcecp principal modify** 命令将对象创建限额复位到非 0 值。注意，删除主体创建的对象时，创建限额不增加。

使用 **dcecp principal show** 命令查看主体当前的对象创建限额。该命令显示允许主体在当前时间创建的对象总数，即原始限额减去主体创建的对象数。

通用唯一标识和 UNIX 标识

"DCE 安全性服务" 自动将主体、组或组织的主名与 UUID 关联。 UUID 标识对象，这是 UNIX 系统中的 UNIX 号 (UNIX 标识) 执行的功能。（注册表数据库也包含 UNIX 号，但是它们仅用来与 UNIX 程序兼容。）

通常，您不必知道 UUID。它们自动创建并维护。要知道虽然可根据 “DCE 安全性服务” 打印出的名称访问对象，但在内部是通过 UUID 标识所有对象的。如果从注册表删除主体，同时也将删除主体的 UUID。任何主体拥有的对象（文件和程序）都与孤立的 UUID 关联，即不带相应名称的 UUID。这意味着该对象归一个已删除的主体所有。如果该对象以前没有允许其它主体访问，则无法访问该对象。

要解决这个问题，使用带有 **-uuid** 选项的 **dcecp principal create** 命令将 UUID 和名称关联，这样就可以接纳孤立的对象。使用 DCE 控制程序的 **principal create** 命令创建对象时将自动指派 UUID。因此，不能简单地添加新的用户并获取以前使用的 UUID。为实现该目的必须执行带有 **-uuid** 选项的 **dcecp principal create** 命令。

注册表中的 UNIX 号必须在设置注册表特性的号的范围内。为创建或修改帐户在命令行提供 UNIX 号时应该避免小于 100 的号码，因为它们通常是为系统帐户而保留的。

添加并维护主体

使用 **dcecp principal create** 命令创建主体。创建主体帐户前必须已经存在该主体。使用 **dcecp principal create** 命令时必须提供主体的主名称作为变量。此外，还可提供表18 中总结的属性选项。

表 18. 创建主体的属性选项

选项	含义
-fullname *namestring*	可选名称用于更完整地描述主名称。要包含空格，将全名用花括号括起来。缺省值为空。
-uid *integer*	需要的与主体关联的 UNIX 标识。可显式地输入该号或允许自动生成它。如果输入它，则输入的号不能超过使用 **registry modify** 命令创建最大允许的 UNIX 号（**maxuid** 属性），但输入的号可以小于用 **registry modify** 命令为主体设置的小 UNIX 号（**minuid** 属性）。如果允许自动指派该号，它将在小 UNIX 号和最大 UNIX 号范围内。
-quota *quota*	主体可以创建的注册表对象数，称为主体的对象*创建限额*。要允许主体创建无限数量的注册表对象，请输入文本字符串 **unlimited** 设置无限额。要阻止主体创建注册表对象，请输入 0。*quota* 变量缺省值为 **unlimited**。

注: 除了这些标准的主体属性，还可以将 ERA 实例附加到主体来控制 DCE 安全性，如预认证、口令强度和口令生成以及无效登录的处理。参阅第311页的『第31章 创建和维护帐户』 以获取关于这些通常使用的 ERA 的信息。参阅第327页的『第32章 创建与使用扩展注册表属性』 以获取关于 ERA 的常规信息。

添加主体

要将主体s 添加到注册表，请使用 **principal create** 命令。例如，下列样本命令创建带有主名称 **mahler** 和全名 **gustav mahler** 的主体:

```
dcecp> principal create mahler -fullname {gustav mahler} -quota 5
dcecp>
```

在本示例中，缺省的 UNIX 号是自动生成的。注意，因为指派到主体的全名 (**gustav mahler**) 包含空格，所以用花括号括起来。

注意可以使用 **principal create** 命令创建多个主体。要完成该操作，将主体名称用花括号括起来并用空格分隔。例如，要创建主体 **bach**、 **britten**、**mahler** 和 **satie**，可以输入下列命令:

```
dcecp> principal create {bach britten mahler satie}
dcecp>
```

如果创建多个主体，必须允许主体的 UNIX 标识缺省为系统指派的标识。这是因为，如果在命令行包含属性选项，则将该属性值指派到每个主体。例如，下列样本命令创建主体 **bach**、**britten**、**mahler** 并为每个主体指派值为 5 的创建限额。

```
dcecp> principal create {bach britten mahler satie} -quota 5
dcecp>
```

更改主体

可以更改主体的主名称和其它与主体相关的信息。另外，可以将主名称更改为别名或将别名更改为主名称。如果将主名称更改为别名但不将别名更改为主名称，则返回名称的操作将随机选择一个别名。

更改主名称

使用 **dcecp principal rename** 命令更改主名称。输入下列格式的命令:

principal rename *old_name* **-to** *new_name*

其中:

old_name
是要更改的主体主名称。

new_name
是主体新的主名称。

下列示例显示了 **principal rename** 命令，用它将全名从 **mahlar** 更改为 **mahler**:

```
dcecp> principal rename mahlar -to mahler
dcecp>
```

注意，如果更改主名称，则将在包含该主体的组和组织列表中反映该更改。

在非常情况下，当主机登录到 DCE 单元时您正在更改主机的主体名称，则现有的主机凭证将变为无效除非执行额外步骤来将主机凭证更新为新的主体名称。

secval 进程管理主机凭证，它在 DCE 主机上执行安全性功能。通常主机启动后，**secval** 进程将主机记录到 DCE 单元，获取主机凭证并在主机上存储它们。更改主体名称后释放并恢复 **secval** 进程以更新这些凭证。下列命令说明了远程主机 **persephone** 上的这些操作:

```
dcecp> secval deactivate /.:/hosts/persephone/config/secval
dcecp> secval activate /.:/hosts/persephone/config/secval
dcecp>
```

注: 本 DCE 发行版不支持将聚集的主体、组或组织 (pgo) 变为它们本身的子代。例如，不允许下列注册表操作序列:

```
dcecp -c group create foo/bar
dcecp -c group create foo
dcecp -c group rename foo -to foo/xyz
```

更改主体信息

使用 **dcecp principal modify** 命令修改除 UNIX 标识和用户标识外的任何主体信息。下列示例显示了用来将主体 **mahler** 的对象创建限额更改为 10 的 **principal modify** 命令。

```
dcecp> principal modify mahlar -quota 10
dcecp>
```

删除主体和别名

如果删除主体或别名，则系统将自动删除主体或别名的任何帐户。例如，如果删除主体**mahler** 也将删除 **mahler composers classic** 帐户。如果删除主体别名 **gustav**，也将删除 **gustav music_admin classic** 帐户。如果删除组别名 **music_admin**，也将删除 **gustav music_admin classic** 帐户。应该知道删除主体或主体别名可能孤立主体/UUID 拥有的对象。

下列示例显示了用来删除名为 **mahler** 的主体的 **principal delete** 命令。

```
dcecp> principal delete mahler
dcecp>
```

可使用 **principal delete** 命令来删除主体或别名。要完成该操作，用花括号将主体名称括起来并用空格分隔。例如，要删除主体 **bach**、**britten** 和 **mahler** 可以输入下列命令:

```
dcecp> principal delete {bach britten mahler}
dcecp>
```

主体的扩展安全性属性

可以将 ERA 实例附加到主体来管理某些 DCE 登录和口令安全性。ERA 可用于控制:

- 主体登录请求所需要的认证级别
- 无效登录处理
- 主体的口令强度和主体的口令生成
- 主体用已失效口令进行登录尝试的处理

在下列部分中将介绍并说明这些 ERA。参阅第327页的『第32章 创建与使用扩展注册表属性』以获取关于如何使用 **dcecp** 命令将这些 ERA 附加到主体的信息。

DCE 认证

认证在 Kerberos V5 认证协议中寻址某些安全性不足，并用作 DCE 版本 1.1 之前版本中的 DCE 认证协议的基础。导致这些不足的因素如下:

- 安全性服务器不验证用户是否知道口令就响应客户机登录请求
- 使用太普遍的弱用户口令来加密跨越网络发送的纯文本数据

这些习惯可能招致攻击，攻击者获取网络传输并脱机继续攻击它们以得到用户口令。如果这些攻击成功，就可能泄露 DCE 单元的安全性（以及与该单元存在信任关系的其它所有单元的安全性）。

DCE 认证通过提供下列措施以减少遭受这些攻击的可能性:

- 对于发出登录请求的主体进行预认证（即通过 “DCE 安全性服务” 在响应请求之前验证请求者的身份）

- 使用强壮密钥加密所有涉及安全性客户机与服务器间确认的网络传输

可使用 5 个级别的认证，从最高级别到最低级别表示严格性递减的预认证协议。通过将 *pre_auth_req* ERA 实例（在以下部分中描述）附加到主体，管理员可以控制认证主体时安全性服务器将接受的最小预认证级别。

预认证协议是:

- *公用密钥证书*协议，它允许 DCE 用户使用 X.509v3 证书和它的关联公用密钥对（来自第三方公用密钥底层设施）来证明它们的身份。要启用公用密钥证书登录，请参阅第289页的『公用密钥证书登录』。
- *公用密钥*协议，它允许 DCE 用户使用通过 DCE 注册表生产并存储在其中的公用密钥来证明它们的身份。该协议已被公用密钥证书协议替代。参阅『启用公用密钥认证协议』 以获取详细信息。
- *第三方*协议，它提供高级别安全性。对于任何主体不应指定更低的预认证级别，除非有强制原因需要这样做。（请参阅下一个公布的条款中 **cell_admin** 命令的注释。）DCE 版本 1.1 客户机总是使用该协议构造认证请求，除非因为需要用来构造第三方请求的机器会话密钥不可用而使它们无法使用该协议。（例如在启动单元时或 **secval** 进程关闭后）。
- *时间戳*协议，它提供中间级别的安全性。应该仅为主体指定时间戳预认证，（例如单元管理员和非交互主体），它们在客户机无法构造第三方认证请求时（如前所述）仍能操作。

  在这些情况下，客户机构造并转发时间戳登录请求。

  特别是单元管理员*必须*具有时间戳登录能力，因为在初始单元配置期间 **cell_admin** 必须能够登录来设置初始机器密钥。
- DCE 版本 1.0 (Kerberos V5) 协议，仅使用它来认证 DCE 版本 1.1 之前的版本，它不提供预认证安全性。

启用公用密钥认证协议

注: DCE AIX 和 Solaris 版客户机不再支持使用 “OSF DCE 1.2.2 公用密钥” 协议的认证。该协议已被 “公用密钥证书协议” 替代。

DCE AIX 和 Solaris 版客户机允许管理员使用与 “OSF DCE 1.2.2 公用密钥” 关联的 **dcecp** 命令来管理公用密钥用户。但是包含用户专用密钥的文件格式是特定于计算机平台的。必须在与要使用它的系统的同类系统上生成它。

DCE AIX 和 Solaris 版安全性服务器支持 “OSF DCE 1.2.2 公用密钥” 协议。它将为基于 OSF DCE 1.2.2 的客户机公用密钥请求服务并支持该功能。

缺省情况下，禁用公用密钥登录认证。要为用户启用并配置该认证，请遵循下列步骤:

1. 确保在主 DCE 安全性服务器和副本服务器上安装了公用密钥软件。对于 DCE 版本 1.2.2 和更高版本可使用公用密钥软件。
2. 初始化单元的公用密钥环境。如果正在将单元从 DCE 1.1 迁移到 DCE 1.2.2，请遵循 2a 和 第289页的2b。否则遵循 第289页的2b。
   a. 启用主服务器上的公用密钥:

   ```
   dcecp> registry modify -version secd.dce.1.2.2
   dcecp>
   ```

主服务器将向每个安装了公用密钥软件的副本服务器传播版本信息。未安装公用密钥软件的任何副本服务器将自动关闭。

b. **cell_admin** 请求为本地单元的 **krbtgt** 主体的帐户生成初始密钥对（公用和专用密钥）：

```
dcecp> account modify krbtgt/cell-name -pkgenprivkey  module size\
> -mypwd pwd
dcecp>
```

注册表将生产密钥对并存储它们。将密钥对中的公用密钥部分存储到附加到该单元的 **krbtgt** 主体的 **DCEPKAuthentication** ERA 中。在现有的口令保护加密方式下将专用密钥部分存储到注册表中。

该发行版支持 512 到 1024 范围内的*模数大小*。虽然用于生成公用密钥对的最小模数大小是 512，但对于大多数用户来说建议使用 768 位作为生成密钥对的模数大小。对于长期的应用程序，可采用 1024 位模数。

注意还可以使用该命令修改本地单元的 **krbtgt** 主体的现有密钥对。支持该单元密钥对 **krbtgt/***cell-name* 仅有的模数大小为缺省值 1024。

3. 在单元管理员工作站上配置用户公用密钥帐户:

```
dcecp> account create name \
> -group group name \
> -mypwd cell-admin-password \
> -organization org_name \
> -password user-password \
> -pkmechanism file \
> -pkkeycipherusage {{generatekey modulus-size} {newpassphrase string}} \
> -pksignatureusage {{generatekey modulus-size} {newpassphrase string}}
```

该命令对于公用密钥帐户完成下列操作:

- 表示应该将专用密钥信息存储在文件中。
- 随机生成新的公用密钥对，它带有为加密目的而提供的附加的新的口令短语。在 ERA 中存储密钥对的公用密钥部分，在文件中存储专用密钥部分。
- 随机生成新的带签名的密钥对，它带有提供的附加的新的口令短语。还是在 ERA 中存储公用密钥部分，在文件中存储专用密钥部分。

4. 将公用密钥机制文件从单元管理员工作站迁移到用户工作站并将所有权更改为该用户。向该用户提供新的口令短语，但强烈建议用户更改口令短语以防止别人知道。要更改口令短语，该用户可以调用 **dcecp account modify** 命令。

```
dcecp> account modify name \
> -pkkeycipherusage {{oldpassphrase string} {newpassphrase newstring}} \
> -pksignatureusage {{oldpassphrase string} {newpassphrase newstring}}
```

缺省情况下，只有单元管理员能够创建或修改公用密钥帐户。但是，单元管理员可以修改公用密钥 ERA 的 ACL 管理器，这样可以允许用户更改它们自身的密钥对。要修改 ACL 管理器:

```
dcecp> xattrschema modify /.:/sec/xattrschema/DCEPKAuthentication \
> -aclmgr {principal {query r} {update u} {test r} {delete m}}
dcecp> xattrschema modify /.:/sec/xattrschema/DCEPKKeyEncipherment \
> -aclmgr {principal {query r} {update u} {test r} {delete m}}
```

公用密钥证书登录

DCE 安全性客户机使用公用密钥预认证协议为用户获取“授予票券的票券”(TGTs)。OSF-RFC 68.4 将 DCE 1.2.2 公用密钥协议扩展允许 DCE 用户使用 X.509v3

数字证书来向 DCE 认证服务证明它们的身份。下列信息描述了基于 OSF-RFC 68.4 的“公用密钥证书登录”的 DCE 3.2 AIX 和 Solaris 版实现。该实现需要“Entrust 公用密钥基础设施”(PKI)。

公用密钥登录概述： DCE AIX 和 Solaris 版允许 DCE 用户使用 X.509v3 数字证书及其关联的公用密钥对（不是共享密钥口令）来证明它们的身份。这种认证机制的一个优点是：如果发生“DCE 安全性服务器”泄密，公用密钥用户将不会对非信任者暴露任何标识信息。使用共享密钥认证，可以向非信任者显示所有用户密钥。另一个即时优点是：由于公用密钥 cryptographic 方法的功效而使基本认证流程更加安全。

该增强是为那些当前使用 Entrust PKI 并需要将 Entrust 用户映射成 DCE 用户从而获得认证并访问 DCE 提供的资源的用户而设计的。该更改仅适用于获取初始的 TGT。一旦接收到了 TGT，则在传统的方式下发生获取附加的服务票券。

除了对 DCE 客户机和“DCE 安全性服务器”交换的信息所作的更改，该增强将提供新服务器“身份映射服务器”(IDMS)。“DCE 安全性服务器”调用该服务器将用户的数字证书映射成 DCE 主体名称。图53 说明了通过该增强建立的认证流程。

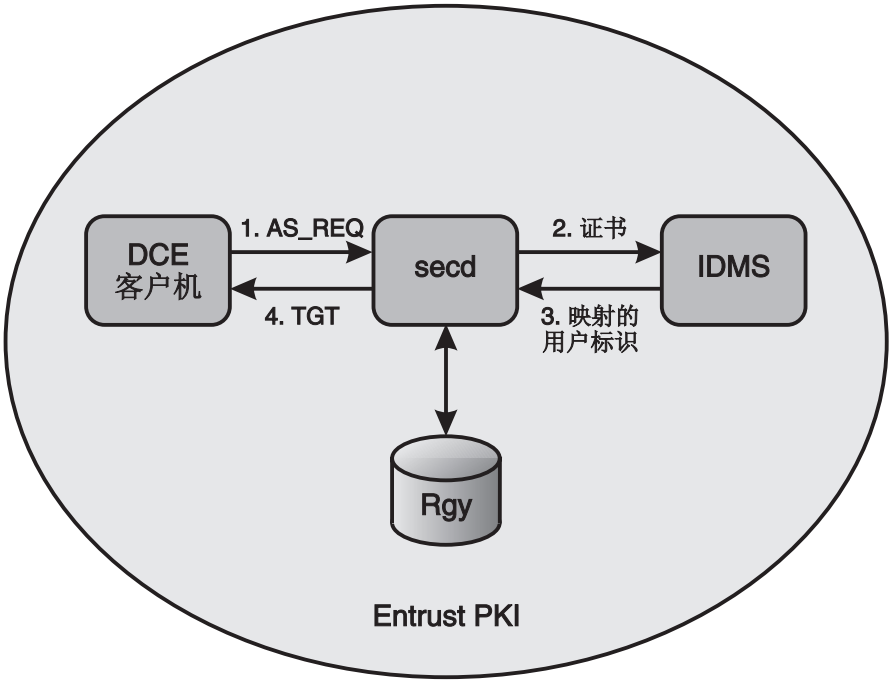


图 53. 公用密钥证书认证流程

使用公用密钥证书登录的 Entrust 先决条件： 该增强需要 Entrust PKI。另外，DCE 客户机和安全性服务器系统必须已经安装了 Entrust 客户机。

对于该增强的每个用户，Entrust 管理员必须创建 Entrust 用户并向每个用户发出公用密钥对以签名并加密。将公用密钥存储到由 “Entrust 认证中心”(CA) 数字签名的公用密钥证书中并存储在 Entrust 用户概要中。在 Entrust 用户概要中存储专用密钥并由口令短语保护（请参阅 Entrust 文档中的口令）。与用户关联的 Entrust 概要必须驻留在用户的客户机系统上。

另外，必须将与每个 Entrust 用户关联的 DCE 主体添加成用户的 X.500 目录项的属性。Entrust 需要 X.500 目录并通过它们的“X.500 特异名称”(DN) 跟踪它的用户。Entrust

提供了创建并管理 Entrust 用户的工具但不与现有的 DCE 工具集成。参考 Entrust PKI 提供的文档以获取有关创建 Entrust 用户的详细的信息。

Entrust 管理员必须还要创建一个“DCE 安全性服务器”使用的 Entrust 用户。单元中的所有安全性服务器都能使用该 Entrust 用户的证书和相关的公用密钥对，或者可为每个“DCE 安全性服务器”创建单独的 Entrust 用户。Entrust 用户概要必须驻留在每个“DCE 安全性服务器”上。IDMS 将使用该系统上安全性服务器正在使用的同一 Entrust 用户概要。

启用公用密钥证书登录： 需要更改两个“DCE 安全性服务器”来启用“公用密钥证书登录”。首先，已将 DCE 配置工具更新成支持附加 IDMS 并允许“DCE 安全性服务器”使用 Entrust 用户概要的位置规范。其次，已经创建了 DCE 注册表 1.2.2a 的新版本。

只有当该单元的“主安全性服务器”运行在版本 1.2.2a 时才启用 DCE 单元的“公用密钥证书登录”。版本 1.2.2 是使用 DCE 3.2 AIX 和 Solaris 版配置的单元的缺省值。一个已从以前的 DCE 版本迁移的单元将继续运行迁移前它正在运行的安全性服务器版本。

应该使用 **dcecp registry modify** 命令通过修改注册表版本号来启用“公用密钥证书登录”。

```
dcecp> registry modify -version secd.dce.1.2.2a
dcecp>
```

警告: 如果单元中的任何“安全性副本”正在运行不支持 “1.2.2a 安全性服务器版本”的 DCE 版本时发出该命令，则将自动关闭副本。

请参阅 *IBM DCE Version 3.2 for AIX: Quick Beginnings* 或 或 *IBM DCE 版本 3.2 Solaris 版: 快速入门* 中的“迁移”部分，以获取关于将现有的 DCE 单元迁移成 DCE 3.2 AIX 和 Solaris 版的详细信息。

在使用“公用密钥证书登录”的 DCE 单元中至少需要一个 IDMS。理想情况下，应该在单元中的每个安全性服务器上都配置 IDMS。下表详细说明了命令行配置选项。 **SMIT** 也支持这些选项并且在配置“ DCE 安全性服务器”或“身份映射服务器”时可以使用它们。

- 配置机器上带有 IDMS 的“安全性服务器”（主服务器或副本服务器）时启用“公用密钥证书登录”，指定 **sec_srv or sec_rep** 组件（适当的）和 **idms_srv** 组件，并且包含下列选项:

  -certificate_based_login yes

  -kdc_ini_file *kdc_ini_file*

  -kdc_profile *kdc_profile*

  -kdc_passphrase *kdc_passphrase*
- 要配置可使用“公用密钥证书登录”但不带 IDMS 的“安全性服务器”，请省略上面的示例中的 **idms_srv** 组件选项。
- 要启用已经配置好的“安全性服务器”上的“公用密钥证书登录”，请指定下列选项:

  -certificate_based_login yes

  -kdc_ini_file *kdc_ini_file*

  -kdc_profile *kdc_profile*

-kdc_passphrase *kdc_passphrase*

只要安全性服务器的版本是 1.2.2a，则下次启动 **secd** 时就可使用“公用密钥证书登录”支持。

- 要将 IDMS 添加到一个已经配置好但不支持“公用密钥证书登录”的“安全性服务器”时，请用下列选项指定 **idms_srv** 组件选项:

  -certificate_based_login yes

  -kdc_ini_file *kdc_ini_file*

  -kdc_profile *kdc_profile*

  -kdc_passphrase *kdc_passphrase*

- 要将 IDMS 添加到一个已经配置好但不支持“公用密钥证书登录”的“安全性服务器”时，只要指定 **idms_srv** 组件选项。

有关这些配置选项的更多信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference*。

使用公用密钥证书登录认证: 没有更改 DCE 登录接口，但是已更改了某些输入值的意义。使用“公用密钥证书登录”，可通过提供您的 Entrust 用户概要而不是 DCE 主体名称来建立您的身份。需要指定用来解锁存储在用户概要中专用密钥的口令短语替代 DCE 口令。

受该增强影响的 DCE API 是 **sec_login_validate_identity( )** 和 **sec_login_valid_and_cert_ident( )**。这些 API 没有附加的标志或变量，它们对现有的变量作出不同解释。这允许现有的登录实用程序和特定平台的集成登录机制认证到带有 Entrust 公用密钥证书的 DCE 而不需重写或重新编译。

例如，如果 Entrust 概要是 **henry.epf** 且您的口令短语是 **Rottweiler9**，则可以使用下列命令登录 DCE:

```
dce_login henry Rottweiler9
```

Entrust 概要和口令短语和 Entrust 客户机的 **entrust.ini** 文件一起允许 DCE 例程调用 Entrust **ETLogin( )** API。这是今后通过 Entrust 的所有公用密钥签名、认证加密与解密操作的基础。这些操作的目的是构造并处理用来在登录 DCE 期间确认用户的预认证数据。

将 Entrust 概要和口令短语作为如前所述的安全性登录 API 的变量进行传递。至 **entrust.ini** 文件的完整路径可从 **.entrustrc** 文件获取。配置 Entrust 客户机时在主目录中创建该文件。该文件中的 ENTRUSTDIR 项给出了 **entrust.ini** 文件的完整路径名。可通过搜索 **.entrustrc** 文件中最近使用的概要来获取 Entrust 概要的完整路径名。如果没有找到，将搜索在 **entrust.ini** 文件中指定的 *DefaultProfileLocation*。

返回传统的认证: 登录接口结构允许在使用“公用密钥证书登录”失败时“返回”共享密钥登录（即使用传统的 DCE 口令）。这是因为 DCE 安全性客户机运行时在向 “DCE 安全性服务器”发出认证请求时将构建并发送公用密钥证书登录协议和 DCE 第三方协议或 DCE 时间戳协议的预认证数据。如果选择使用 DCE 主体名称命名 Entrust 用户概要并且您的 Entrust 口令短语与传统的 DCE 口令匹配，则返回对您是透明的。*pre_auth_req* ERA 用来确定是否允许返回。

身份映射服务器: “身份映射服务器”(IDMS) 是将用户的公用密钥证书映射成 DCE 主体名称的新 RPC 服务器。在使用“公用密钥证书登录”的 DCE 单元中至少需要一个

IDMS。理想情况下，应该在使用“公用密钥证书登录”的 DCE 单元中每个“安全性服务器”上都配置 IDMS。用户使用公用密钥证书认证登录 DCE 时“DCE 安全性服务器”调用 IDMS。IDMS 支持将许多 Entrust 用户映射成一个 DCE 用户以及更传统的一对一用户映射的需要。

IDMS 需要将与 Entrust 用户关联的 DCE 主体添加为用户 X.500 目录项的附加属性。Entrust 需要 X.500 目录并通过它们的 X.500 DN 跟踪其用户。具体地说必须将名为 **dcePrincipal** 的属性添加到表示 Entrust 用户的对象目录模式中。Entrust 用户的典型类型是 **organizationalPerson** 和／或 **entrustUser** 这两种类型。然后使用用户证书中的 X.500 DN 完成从 Entrust 用户到 DCE 主体的映射，并执行目录查询以查找用户的 **dcePrincipal** 属性。请参阅 “X.500 目录服务”提供的文档以获取关于添加属性的附加信息。

在这种方式下执行身份映射，因为它能使“DCE 安全性服务器”更加灵活地调用单独服务器以获取映射。这样就可能定制映射算法。因为身份映射策略将根据个别的用户需求而发生变化，提供缺省 IDMS 的源代码作为 DCE 示例程序。该示例程序安装在 **/usr/lpp/dce/examples/idms** 目录中。

公用密钥证书登录的限制： 对于配置使用公用密钥认证的帐户有几条限制。它们包括:

- 不允许保留的 DCE 主体使用“公用密钥证书登录”。在 DCE 3.2 AIX 和 Solaris 版中，设置了保留标志的主体包括:
  - 初始配置单元时指定的管理标识
  - 单元的 krbtgt 主体
  - 初始主安全性服务器系统自身的主体

  该限制的副作用是无法提供 Entrust 用户概要名称作为 DCE 配置期间的 cell_admin 标识。
- 无法使用 **kinit** 命令刷新已失效的 DCE 凭证除非提供 DCE 口令。该刷新操作不支持使用 Entrust 用户概要和口令短语。如果 Entrust 用户概要名称和口令短语与 DCE 主体名称和口令同步，则该限制对于用户是透明的。
- 将多个 Entrust 用户映射成单一 DCE 主体时将减小 DCE 的具体功能（例如审计和访问控制）级别。只能使用 DCE 主体信息，用于审计记录和访问控制检查。
- 如果将需要口令强度检查的 *pwd_val_type* ERA 附加到 DCE 主体，则只对该主体的 DCE 口令强制进行这些检查。Entrust PKI 建立对于 Entrust 口令短语强制执行的一组单独规则。
- 只有使用共享密钥认证协议的应用程序才使用密钥管理 API。使用公用密钥证书登录的应用程序必须使用用户到用户协议。
- 使用 GSSAPI 时 DCE 管理员必须为启动程序和接收程序在 DCE 注册表数据库中设置帐户。接收程序无法使用“公用密钥”认证。对于启动程序的帐户没有限制。
- DCE 3.2 AIX 和 Solaris 版中的“PKC 登录”支持是以 IETF RFC 1510 - Kerberos 中的初始认证公用密钥加密术 (Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)) 1998 年稿为基础的。因为该稿未最终定稿，所以还无法保证 PKINIT 将来版本的互操作性。
- **dce_login -r** 命令无法用来刷新已失效的 DCE 凭证除非提供了 DCE 口令。不支持 Entrust 用户概要和该刷新操作的口令短语的使用。
- （**仅 AIX**）此时不支持使用带有 AIX/DCE 集成登录选项的“PKC 登录”。

Entrust 需求与限制: 只有当计划使用“公用密钥证书登录”特性是才需要 Entrust 产品。在 AIX 和 Solaris 上，该特性需要下列:

- 每个允许“PKC 登录”的“DCE 客户机”上的 Entrust/Toolkit 版本 5.0.2 “Entrust 引擎”库。
- 每个“DCE 安全性服务器”和“身份映射服务器”上的 Entrust/Toolkit 版本 5.0.2 “Entrust 引擎”库。

Entrust/Toolkit 版本 5.0.2 包含改正在测试“公用密钥证书登录”特性期间发生的问题的补丁程序。Entrust/Toolkit 的该补丁级别可从 Entrust Technologies FTP 站点获取。

DCE 系统不需要“Entrust 公用密钥基础设施”，但可使用它向用户发出证书。推荐的 Entrust/PKI 级别是版本 5.0。

测试“PKC （公用密钥证书）登录”支持时将遇到下列 Entrust/Toolkit 限制:

- 在带有固定数量的当前登录活动环境下，“DCE 安全性服务器”在处理认证消息时可能从 Entrust 接收到错误。这将导致“PKC 登录”尝试失败。如果启动 **secd** 命令前在“DCE 可服务性”路由文件中启用 NOTICE 级别记录，则“DCE 安全性服务器”将记录 **notice.log** 文件中的 Entrust 错误。
- 不支持在使用“PKC 登录”的 *lite 方式*中使用 Entrust/lite 或 Entrust/PKI。

使用“公用密钥证书登录”所需的步骤摘要: 下表总结了使用“公用密钥证书登录”增强所要采取的步骤。

1. 配置“Entrust 公用密钥基础设施”并创建 Entrust 用户。
2. 安装 Entrust 客户机的系统是或将是 DCE 客户机或“安全性服务器”。
3. 将 Entrust 用户概要放置到 DCE 客户机和“安全性服务器”系统上。
4. 将 DCE 单元配置或迁移到注册表版本 1.2.2a。
5. 在单元中每个“安全性服务器”上配置 IDMS 和“公用密钥证书”选项。
6. 将 **dcePrincipal** 属性添加到用户的 X.500 目录项中。
7. 用户可登录并指定它们的 Entrust 概要名称和保护其概要的口令短语。

管理 DCE 用户认证

可通过将 *pre_auth_req* ERA 实例附加到用户的主体并指定一个值来表示“DCE 安全性服务”应该接受主体的*最低级别协议*来管理给定 DCE 用户的预认证，如下所示:

0 **（零）** 指定“DCE 安全性服务”应该接受从使用这 4 种协议中任何一种的主体发出的登录请求（包括 DCE 版本 1.1 之前版本的协议）。这是最低安全级别且仅提供给 DCE 版本 1.1 服务器以接受从 DCE 版本 1.1 之前版本客户机发出的登录请求。它最容易受到前面描述的那种类型的攻击。

警告: 如果将 *pre_auth_req* ERA 附加到主体时失败则等价于指定 **0（零）**。

1 **(PADATA-ENC-TIMESTAMPS)** 指定“DCE 安全性服务”应该接受从使用时间戳协议、第三方协议或公用密钥协议的主体发出的登录请求。时间戳协议将防范装扮成安全性客户机的攻击者和从“DCE 认证服务”发出的攻击性回答。该协议仍然容易遭受来自能够监控网络的进程的攻击。

2 **(PADATA-ENC-THIRD-PARTY)** 指定“DCE 安全性服务”只接受从使用第三方协议或公用密钥协议的主体发出的登录请求。该协议提供高级别 DCE 预认证

并提供攻击保护。使用第三方预认证，所有跨越网络发送的预认证数据都将使用只有本地机器主体和“DCE 安全性服务”知道的随机强壮密钥进行加密。

3 **(PADATA-ENC-PUBLIC-KEY)** 指定“DCE 安全性服务”只接受从那些使用公用密钥协议的主体发出的登录请求。

4 **(PADATA-ENC-PUBLIC-KEY-CERTIFICATE)** 指定“DCE 安全性服务”只接受从那些使用公用密钥证书登录协议的主体发出的登录请求。

DCE 认证服务接收到主体的登录请求时，它总是试图使用与请求相同的协议响应，除非该主体的 *pre_auth_req* ERA 值禁止它这样做。 表19 提供的矩阵描述了认证服务在登录（认证）请求类型和 *pre_auth_req* ERA 值的各种组合下采取的操作。

关于 DCE 认证的完整详细信息（包括认证协议的操作），请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Application Development Guide--Core Components*。

下列是使用 **dcecp** 命令修改主体并附加指定需要“公用密钥证书登录”的 *pre_auth_req* ERA 的示例:

```
dcecp> principal modify name -add {pre_auth_req 2}
dcecp>
```

如果允许用户使用“公用密钥证书登录”但是不需要使用它时没有必要设置 *pre_auth_req* ERA。关于如何使用 **dcecp** 命令将 ERA 附加到主体的进一步信息，请参阅第327页的『第32章 创建与使用扩展注册表属性』。

Entrust 消息

没有将“Entrust 公用密钥”基础设施提供的消息翻译成 DCE 支持的所有语言。使用“公用密钥”证书登录时， DCE 非英文版本有时会显示英文消息。

DCE 版本之间的公用密钥互操作性

表19 描述了在单一单元中如何处理 DCE 不同版本间的登录请求。本表只包含“服务器版本 1.1”或更高版本，因为 1.1 版之前的“服务器”总是忽略登录请求中的预认证数据并返回 DCE 版本 1.1 之前版本（未预认证的）的响应。

表 19. DCE 认证互操作

登录请求类型	版本 1.1 和 1.2 服务器响应	版本 1.2.2a 服务器响应
DCE 版本 1.0		
从任何客户机。	预认证。检查 *pre_auth_req* ERA 实例: 如果不存在 ERA 或现有的 ERA 为 *value*=**0**（零），将返回 DCE 版本 1.0 （未预认证的）响应。否则，拒绝登录请求。	预认证。检查 *pre_auth_req* ERA 实例: 如果不存在 ERA 或现有的 ERA 为 *value*=**0**（零），将返回 DCE 版本 1.0 （未预认证的）响应。否则，拒绝登录请求。
时间戳		

表 19. DCE 认证互操作 (续)

登录请求类型	版本 **1.1** 和 **1.2** 服务器响应	版本 **1.2.2a** 服务器响应
从 DCE 版本 1.1 和更高版本客户机。	预认证。检查 *pre_auth_req* ERA 实例: 如果不存在 ERA 或现有的 ERA 为 *value*=**0** (**零**) 或 *value*=**1** (**PADATA-ENC-TIMESTAMPS**), 返回 DCE 版本 1.1 **时间戳**响应。如果现有的 ERA 为 *value*=**2** (**PADATA-ENC-THIRD-PARTY**), 将拒绝登录请求。 1.2 服务器响应: 如果 ERA 为 *value*=**3** **(PADATA-ENC-PUBLIC-KEY)**, 也将拒绝登录请求。	预认证。检查 ERA 实例: 如果不存在 ERA 或现有的 ERA 为 *value*=**0** (**零**) 或 *value*=**1 (PADATA-ENC-TIMESTAMPS)** 将返回 DCE 版本 1.1 时间戳响应。如果现有的 ERA 为 *value*=**2** **(PADATA-ENC-THIRD-PARTY)**、*value*=**3** **(PADATA-ENC-PUBLIC-KEY)** 或 *value*=**4** **(PADATA-ENC-PUBLIC-KEY** **-CERTIFICATE)**, 将拒绝登录请求。
第三方		
从 DCE 版本 1.1 和更高版本客户机。	1.1 服务器响应: 预认证。返回 DCE 版本 1.1 **THIRD-PARTY** 响应。 1.2 服务器响应: 预认证。检查 *pre_auth_req* ERA 实例: 如果存在 ERA 且为 *value*=**3** **(PADATA-ENC-PUBLIC-KEY)**, 将拒绝登录请求。否则返回 **THIRD-PARTY** 响应。	预认证。检查 *pre_auth_req* ERA 实例: 如果存在 ERA 且为 *value*=**3** **(PADATA-ENC-PUBLIC KEY)** 或 *value*=**4** **(PADATA-ENC-PUBLIC KEY** **-CERTIFICATE)**, 将拒绝登录请求。否则返回 **THIRD-PARTY** 响应。
公用密钥		
从 DCE 版本 1.2.2 客户机 (不包括 IBM DCE Windows NT® 版, 版本 2.2 客户机) 和 DCE AIX 版, 版本 2.3 客户机, DCE 级别 2.2.0.4 或更高级别 (请与您的服务代表联系以获取实际 APAR 号)。	1.1 服务器响应: 预认证。返回 DCE 版本 1.1 **THIRD-PARTY** 响应。 1.2 服务器响应: 预认证。返回 DCE 版本 1.2.2 **PUBLIC-KEY** 响应。	预认证。检查 *pre_auth_req* ERA 实例: 如果存在 ERA 且为 *value*=**4** **(PADATA-ENC-PUBLIC-KEY** **-CERTIFICATE)**, 将拒绝登录请求。否则, 返回 DCE 版本 1.2.2 **PUBLIC-KEY** 响应。
共用密钥认证		
从 IBM DCE Windows NT 版, 版本 2.2 客户机 或 DCE Solaris 版, 版本 2.3 客户机 或 DCE AIX 版, 版本 2.3 或 2.2 客户机, DCE 级别 2.2.0.4 或更高级别 (请与您的服务代表联系以获取实际的 APAR 号)。	1.1 服务器响应: 预认证。返回 DCE 版本 1.1 **THIRD-PARTY** 响应。 1.2 服务器响应: 预认证。检查 ERA *pre_auth_req* 实例: 如果存在 ERA 且为 *value*=**3 (PADATA-ENC-PUBLIC-KEY)**, 将拒绝登录请求。否则返回 **THIRD-PARTY** 响应。	预认证。返回 **PUBLIC-KEY-CERTIFICATE** 响应。

管理无效登录处理

将主体的预认证级别指定为 **2** (**PADATA-ENC-THIRD-PARTY**) 时安全性服务器就能够检测并跟踪该主体的无效登录尝试。这就使管理员可能通过下列措施限制可能的口令猜测攻击:

- 设置禁用的主体帐户前连续的无效登录尝试的数量限制。(成功登录复位计数器)。

- 指定一旦到达限制值禁用主体帐户的时间周期。

通过将两个 ERA 实例（*max_invalid_attempts* 和 *disable_time_interval*）附加到主体中来完成该操作。为这些 ERA 指定下列值:

max_invalid_attempts
指定一个整数，它表示将主体帐户标记为禁用前安全性服务器应该接受的连续无效登录试图数量。

disable_time_interval
指定一个整数，它表示从登录尝试开始禁用主体帐户的秒数。

下列是使用 **dcecp** 命令创建主体并附加 *max_invalid_attempts* 和 *disable_time_interval* ERA 的示例:

```
dcecp> principal create smitty -attribute {{max_invalid_attempts
7} \
> {disable_time_interval 60}}
dcecp>
```

注: 在 DCE 版本 1.1 上无效登录处理功能在带有主服务器但无副本的单元中准确地跟踪登录活动，但不在副本单元中准确计数。这是因为:

- 将副本单元中的登录尝试随机指派到主服务器或副本。
- 目前没有副本与主服务器通信的机制，所以主服务器无法维持准确计数。

关于如何使用 **dcecp** 命令将 ERA 附加到主体的进一步信息，请参阅第327页的『第32章 创建与使用扩展注册表属性』。

管理口令强度和口令生成

第353页的『第35章 维护策略和特性』中描述的 DCE 口令格式策略将使您能够控制用户口令的下列特征:

- 最小口令长度
- 口令是否可全部为空
- 口令是否可以只能包含字母数字字符

通过创建口令管理服务器来执行定制的口令检查和生成，在您的单元扩展这些口令强度策略。 DCE 提供口令验证／生成服务器示例 **pwdstrn2**，可以使用它作为适合单元需求的口令管理服务器的基础。DCE 还提供了口令强度服务器 **pwd_strengthd**。

DCE 提供了“口令管理 API”，应用程序开发商可以使用它获取关于主体口令管理策略的信息，并从口令管理服务器请求已生成的口令。参阅 *IBM DCE Version 3.2 for AIX and Solaris: Application Development Guide--Core Components* 以获取关于“口令管理 API”的信息。

创建该服务器后，然后可以在创建该口令以及无论何时更改它后强制服务器确认主体口令。可通过将 *pwd_val_type* 和 *pwd_mgmt_binding* ERA 实例附加到主体来实现该操作，如下:

pwd_val_type
指定主体的口令创建选项如下:

0 （零） 指定主体口令只能遵循 DCE 标准策略。（参阅第353页的『第35章 维护策略和特性』 以获取关于 DCE 标准策略的描述。）指定 **0**（零）与不将 ERA 实例附加到主体等价。

1 **(USER_SELECT)** 指定主体必须提供口令文本作为 *pwd_mgmt_binding* ERA 中指定的口令管理服务器的输入。

2 **(USER_CAN_SELECT)** 指定主体提供口令或选择通过 *pwd_mgmt_binding* ERA 中指定的口令管理服务器提供的已生成口令。

3 **(GENERATION_REQUIRED)** 指定 *pwd_mgmt_binding* ERA 中指定的口令管理服务器应该为主体生成口令。

pwd_mgmt_binding

指定您单元的口令管理服务器的绑定。下列术语组成 *pwd_mgmt_binding* ERA:

认证服务

指定认证服务类型。总是使用 **dce**。

口令强度服务器主体

指定口令强度服务器主体的名称。使用为 **-server_princ** 选项指定的相同名称。缺省值是 **pwd_strengthd**。

保护级别

指定使用该绑定的远程过程调用的保护级别。保护级别是 **pktprivacy**、**cdmf** 和 **pktinteg**。

认证级别

指定认证级别。总是使用 **secret**。

认证服务

指定通过服务器实现的认证服务。总是使用 **name**。

cds 对象名

指定绑定。

下列是 **dcecp** 命令创建主体并附加 *pwd_val_type* 和 *pwd_mgmt_binding* ERA 的示例:

```
dcecp> principal create smitty -attribute {{pwd_val_type 2} \
> {pwd_mgmt_binding \
> {{dce pwd_strengthd pktprivacy secret name} \
> {/.:/subsys/dce/pwd_mgmt/pwd_strengthd}}}
dcecp>
```

关于如何使用 **dcecp** 将 ERA 附加到主体的进一步信息，请参阅第327页的『第32章 创建与使用扩展注册表属性』。 有关更改口令时请求已生成的口令的信息，请参阅第299页的『使用 dcecp 生成口令』。

有关配置口令管理服务器的信息，请参阅下列部分和 *IBM DCE 版本 3.2 AIX 和 Solaris 版: 管理指南--简介*。

管理口令管理服务器

*IBM DCE 版本 3.2 AIX 和 Solaris 版: 管理指南--简介*第二部分说明了如何使用 **config.dce** 配置口令管理服务器。该部分提供了口令管理服务器管理的附加的注意事项。

- 要保护口令安全性并优化性能，口令管理服务器应该在与主 DCE 安全性服务器相同的机器上运行。
- 口令管理服务器的缺省路径名为 **opt/dcelocal/bin**。
- **config.dce** 只支持单元中一个口令管理服务器的配置时，可能要手工配置附加服务器。然后可以设置主体*pwd_mgmt_binding* ERA 来指向每个主体的相应的服务器。

- 要用另一版本替换口令管理服务器，请遵循下列过程:
  1. 停止 **pwd_strengthd**。
  2. 重命名 **opt/dcelocal/bin/pwd_strengthd**。
  3. 将新的服务器复制到 **opt/dcelocal/bin/pwd_strengthd** 中。
  4. 启动 **pwd_strengthd**。
- 口令管理服务器的日志文件驻留在 **opt/dcelocal/var/security/pwd_strength** 目录中。日志文件名是从 **-server_princ** 选项指定的名称生成的。如果未指定 **-server_princ** 选项，缺省日志文件名则为 **pwd_strengthd.log**。

使用 dcecp 生成口令

如果主体存在 *pwd_val_type* ERA 的值为 **2** (**USER_CAN_SELECT**) 或 **3** (**GENERATION_REQUIRED**)，则主体可以（或被需要）在更改口令时请求已生成的口令。如果您是主体 **smitty**，则下列 **dcecp** 命令序列可用来完成该操作:

```
dcecp> set p [account generate smitty]
newgenpwd
dcecp> account modify smitty -password $p -mypwd -dce-
dcecp>
```

该命令从口令管理服务器请求一生成的口令，将新口令放置在 *p* 变量处并在屏幕上显示它 (*newgenpwd*)。（请确保记住新口令。）然后，传递存储在 *p* 中的值作为 **account modify** 或 **account create** 命令中新口令的值。

警告: 请不要执行下列 **dcecp** 命令，因为这将在帐户中更改口令，但用户无法看到新生成的口令:

```
dcecp> account modify smitty -password [account gen smitty] -mypwd -dce-
dcecp>
```

使用口令强度服务器更改口令

用户可使用 **dcecp**、**rgy_edit** 或 **dce_login** 命令中的 **-n** 选项来更改口令。如果单元管理员需要复位用户口令，可使用 **rgy_edit** 或 **dcecp** 命令。

在 **dcecp** 命令中，*pwd_val_type* 为 **3**（它需要用户口令通过“口令服强度务器”生成）只采用 2 步过程（在以上的 『使用 dcecp 生成口令』 中定义）就可以更改它们的口令。

dce_login 或 **rgy_edit** 命令将在提示输入新口令前为 *pwd_val_type* 为 **2** 或 **3** 的用户显示已生成口令。

IBM DCE 增强口令强度服务器 (Enhanced Password Strength Server)

“IBM DCE 增强口令强度服务器”执行扩展口令的检查。该服务器可确认违背组合规则、寿命规则、再使用（历史）规则、字典列表和用户定义规则的口令。“增强口令强度服务器”使用 *IBM_pwd** ERA 确定要强制的规则。可将 ERA 在用户定义、organizational 或 server-wide 级别上设置。

帐户能够使用增强的口令验证之前，必须使用 *pwd_val_type* 和 *pwd_mgmt_binding* ERA 将主体定义到口令检查服务器中。参阅 298 页的 **dcecp** 命令示例。一旦主体启用口令强度检查或口令生成或同时启用二者，口令检查服务器可以使用 *IBM_pwd** ERA 确定强制使用何种规则。

注: 以前 DCE 版本中“口令强度服务器”上可使用的某些命令行选项现已过时。这些选项是:

- **+/-all_spaces**
- **+/-alpha_num**
- **-min_len**

虽然为了兼容性目的在增强的服务器上这些选项继续有效，但应该避免使用它们。增强的服务器可从注册表读取类似的口令规则并使用它们检查用户口令。使用 dcecp 命令 **registry modify** 和 **organization modify**，这可通过设置 registry-wide 或组织定义口令策略来实现。

ERA 规则

每个分组的 ERA 类别和属性定义如下:

IBM_pwd_comp_rules ERA: 该“组合检查 ERA”指定在口令中可使用的字符的类型和组合。支持下列规则 :

mindiff
新口令中必须与旧口令有不同的字符的最小数。

maxrepeat
给定字符在口令中出现的最多次数。

minalpha
口令中字母字符的最小数。

minother
口令中非字母字符的最小数。

minlen
口令中字符的最小数。

maxlen
口令中字符的最大数。

使用一系列 attribute=value 对定义规则。

例如:

```
dcecp> principal modify smitty -add {IBM_pwd_comp_rules mindiff=4 maxrepeat=2 \
minalpha=4 minother=3 minlen=7 maxlen=8}
```

注: 在下列情况下，“IBM 增强口令强度服务器”将复位 **maxlen**:

- 如果 **minalpha** 和 **minother** 的组合大于 **maxlen**，则将 **maxlen** 设置成 **minalpha** + **minother**。
- 如果 **minlen** 大于 **maxlen**，则将 **maxlen** 设置成与 **minlen** 相等。

将忽略出现在 ERA 中的无法认证的属性名。

IBM_pwd_age_rules ERA: 该“寿命检查 ERA”提供能够指定多久可更改口令的能力。 支持下列规则 :

minage
更改口令前必须使用该口令的最小时间。最小寿命规则定义更改口令前必须使

用该口令的最小时间。最后修改帐户口令时的时间戳记已存储在“安全性注册表”中。将获取该信息并将它与当前时间相比较以确定是否允许修改该口令。指定的时间是周数。

使用一系列 attribute=value 对定义规则。

例如:

```
dcecp> principal modify smitty -add {IBM_pwd_age_rules minage=1}
```

将忽略出现在 ERA 中的无法认证的属性名。

IBM_pwd_hist_rules ERA: “历史检查”--可再次使用口令前的时间和再次使用口令前必须使用的不同口令数。直到符合一定标准时才能将用户以前的口令作为新口令再次使用。支持下列规则:

histexpire
在能够再次使用以前的口令之前必须经过的已定义的时间段。指定的时间是周数。

histsize
在能够再次使用以前的口令之前必须发生的口令更改数。

注: 为了在历史数据库中保存空间，只保存用来增强历史规则的口令。

注: 对于 **mindiff**、**minage**、**histexpire** 和 **histsize**，新口令与当前口令的比较规则要求在口令强度服务器上的口令更改必须已经在规则成功前发生。检查例程从口令强度历史数据库中检索当前口令。如果未使用口令强度服务器更改当前口令，则它不会存在于历史数据库中。在这种情况下，将新口令与空白比较，该操作可能失败。

使用一系列 attribute=value 对定义规则。

例如:

```
dcecp> principal modify smitty -add {IBM_pwd_hist_rules histexpire=26 histsize=25}
```

将忽略出现在 ERA 中的无法认证的属性名。

IBM_pwd_dict_rules ERA: 该“字典检查 ERA”指定在口令验证期间使用的字典文件列表。字典文件包含不适合用作口令的字列表。例如，管理员可能想防止用户使用这些容易猜出的普通字（例如公司名称）。管理员可能想进一步限制个别用户使用带有个人含义的字（例如家庭成员姓名）。该 ERA 的值是在口令验证中使用的字典文件列表。

例如:

```
dcecp> principal modify smitty -add {IBM_pwd_dict_rules dictionary.english}
```

文件格式: 字典文件与文档化的“AIX 和 Solaris 字典文件”具有相同的格式:

- 每行一个字
- 每个字从第一列开始以新行字符结束
- 不除去嵌入的、开头的和后跟的空白

驻留: 字典文件必须驻留在正在运行口令强度服务器的机器上。可用绝对或相对路径指定字典文件。相对路径是相对于服务器数据目录而言的:

/opt/dcelocal/var/security/pwd_strength

如果将口令强度服务器移到新的机器，请确保传送该字典文件。

丢失的文件: 如果口令强度服务器无法找到指定的文件或无法访问该文件，将记录错误并无法修改口令。

将出现在 ERA 中的未认证的属性名或值记录为错误。

IBM_pwd_userdef_rules ERA: 该“用户定义的规则 ERA”使管理员能够在提供的规则不满足他们需要情况下编写自己的口令检查例程。例如，用户可能要求每个口令的第二个字符必须是数字。该 ERA 的值必须是将要在口令验证中使用的 C 共享对象。

注: 增强的口令强度服务器必须以 **-userdef** 选项开始以执行用户定义的检查。否则，将生成错误信息并且无法修改口令。

例如:

```
dcecp> principal modify smitty -add {IBM_pwd_userdef_rules pw.checker}
```

通过在运行时装入指定的共享对象并向其入口点传递信息来实现根据“用户定义规则”检查口令。入口点必须是带有下列定义的函数:

```
int pwdrestrict_method(char *user, char *new_passwd, char *old_passwd, char **message);
```

所有口令限制方法的约定是将后台消息传递到正在调用的子例程。不要将消息打印到 stdout 或 stderr。注意: 必须将消息返回到动态分配内存以调用程序。该调用程序将在使用内存后释放内存。

关于可装入子例程模块有许多警告:

1. *NewPassword* 和 *OldPassword* 的值实际上是用户输入的纯文本口令。如果将这些口令复制到内存的其它部分，在返回调用程序前清除那些内存占用。这有助于防止纯文本口令在磁心信息转储中显示。另外，不要把这些口令复制到文件或其它程序可访问的任何地方。纯文本口令绝对不应该存在于进程空间以外。
2. 不要修改进程信号处理器的当前设置。
3. 不要调用将终止程序执行的任何函数（例如 **exit** 子例程和 **exec** 子例程）。总是返回正在调用的程序。
4. 代码必须是线程安全的。
5. 实际的装入模块必须保存在写保护的环境中。只有 root 用户可写装入模块和目录。

用户定义的检查方法是由服务器强制使用的最后限制。

参数:

UserName
指定用户名。

NewPassword
指定纯文本格式的新口令。该值可以为“空”指针。

OldPassword
指定纯文本格式的当前口令。该值可以为“空”指针。

Message

指定指向包含错误消息的 **malloc** 内存的指针地址。期望该方法提供 **malloc**'ed 内存和消息。

返回值: 期望该方法返回下列值。返回值以优先权顺序列出。

-1 内部错误。该方法无法执行其口令求值。该方法必须在*消息*中提供错误消息，除非无法为该消息分配内存。如果它无法分配内存，则必须在*消息*中返回“空”指针。

1 失败。口令更改不能满足限制需求。正确评估了口令限制但不接受口令更改。该方法必须在*消息*中提供错误消息。

0 成功。口令更改满足限制需求。

以下注释仅适用于 AIX:

注:

以下告诉编译器和装入器编译输入文件并使用指定的检查例程作为结果目标文件的入口例程:

```
cc -e checking_routine -o out_file input_file
```

例如:

```
cc -e pwdrestrict_method -o pwd_check.libpwd_checker.c
```

请参阅 AIX Version 4.3 Base Operating Systems and Extensions Technical Reference 中的 **pwdrestrict_method Subroutine** 卷 1，以获取有关该函数的详细信息。使用该接口允许已为 “AIX 安全性”编写了“用户定义规则”的用户继续使用那些未修改的规则。

以下注释是特定于 Solaris 的:

注:

以下告诉 c 编译器和装入器编译包含 **pwdrestrict_method routine** 的输入文件并为结果文件创建可装入库。

```
cc -G -o out_fileinput_file
```

例如:

```
cc -G -o pwd_check.lib pwd_checker.c
```

规则定义: “用户定义规则”实现与“字典规则”很相识。使用 ERA (**IBM_pwd_userdef_rules**) 保存在口令检查期间使用的共享对象的文件名。

驻留: 共享文件对象必须驻留在正在运行口令强度服务器的机器上。可用绝对或相对路径指定共享对象文件。相对路径是相对于服务器数据目录 **/opt/dcelocal/var/security/pwd_strength** 而言的。

如果将口令强度服务器移到新的机器，请确保传送该共享对象文件。

丢失的文件: 如果口令强度服务器无法找到指定的共享对象或无法访问该文件，将记录错误并无法修改口令。

将出现在 ERA 中的未认证的属性名或值记录为错误。

用户定义、Organizational-Wide 和 Server-Wide 值

对于用户定义的值，将 ERA 直接附加到用户的主体。对于 organizational-wide ERA，应该将 ERA 附加到组织标识中。应该将 server-wide ERA 附加到口令强度的主体中。这是用 **-server_princ** 选项指定的口令强度服务器的名称。对于定义到口令强度服务器中的任何用户的口令更改，将检查 server-wide 和 organizational-wide 规则。请参阅『合并规则』以获取关于解决为一个用户定义多个规则时的冲突的信息。

server-wide 规则的示例:

```
dcecp> principal modify pwd_strengthd -add {IBM_pwd_comp_rules minlen=8 maxlen=10}
```

注: 使用 **-server_princ** 选项配置口令服务器主体。

organizational-wide 规则示例:

```
dcecp> organization modify area2 -add {IBM_pwd_dict_rules dictionary.english}
```

口令生成

已修改了口令生成例程来执行已生成口令的验证。只在口令高速缓存中存储那些已根据用户、组织和 server-wide 规则成功验证的已生成口令。

如果在 **pwd_mgmt_binding** ERA 中使用 **/.:** 单元名称简写（即 subsys/dce/pwd_mgmt/pwd_strengthed），则跨越内部单元边界的口令生成将无效。该限制防止内部单元连接上带有 2 或 3 **pwd_val_typd** ERA 的主体的口令更改。

要防止该问题发生，请使用完整 canonical 单元名称指定绑定。

例如，可指定:

```
dcecp -c principal modify princ_name -add {pwd_mgmt_binding  {{ dce pwd_strengthd \
pktprivacy secret name}/.../cell_name/subsys/dce/pwd_mgmt/pwd_strengthd}}
```

该命令将显式地将 *princ_name* 定向到 *cell_name* 而不是执行口令更改时缺省本地单元中的“口令强度服务器”。

合并规则

对于“IBM DCE 增强口令强度服务器”，如果为用户、组织或服务器或所有定义了规则，则将增强带有最严格值的规则。最严格的值将按照表20增强:

表 20. 合并规则

规则	最严格
mindiff	最大值
maxrepeats	最小值
minalpha	最大值
minother	最大值
minlen	最大值
maxlen	最小值
minage	最大值
histexpire	最大值
histsize	最大值

注: 将合并字典列表和用户定义的列表；如果是为用户、组织和服务器定义的，则在口令验证期间将使用所有列表。

管理口令失效

缺省情况下，“DCE 安全性服务”禁用那些口令已经失效的主体登录。在许多情况下您不想这样，例如可能不想因为口令失效而将 **cell_admin** 锁定在单元之外。如果 secd 口令失效或自身主体口令失效，secd 或 dced 将无法认证并且不能使用守护进程。

推荐的模式是将这些极限设置为 ″forever″ 并根据组织或帐户进行细化。如果 secd 口令失效或其自身主体口令失效，则进入 secd locksmith 方式并作必要更改。

可通过将 *passwd_override* ERA 实例附加到主体中并指定下列值之一来管理给定主体的口令失效检查:

0 （零）指定不忽略主体的口令失效检查（即不允许使用失效口令的主体登录。）指定 **0**（零）与不将 ERA 实例附加到主体等价。

1 (**OVERRIDE**) 指定将忽略主体的口令失效检查（即应该允许主体使用已失效的口令登录。）

以下是 **dcecp** 命令创建主体并附加 *passwd_override* ERA 的示例:

```
dcecp> principal
create smitty -attribute {passwd_override 1}
dcecp>
```

关于如何使用 **dcecp** 将 ERA 附加到主体的进一步信息，请参阅第327页的『第32章 创建与使用扩展注册表属性』。

添加并维护组和组织

在帐户中使用组或组织之前必须已经将它们添加到注册表中。使用 **dcecp group -create** 命令添加组时可以设置控制是否在项目列表中包含单一组的项目列表包含特性。（项目列表不适用于组织。）

项目列表

主体的项目列表是一个全部组列表，主体或别名是它的成员。主体尝试访问对象时，将在对象的 ACL 中命名拥有从每个组成员资格产生访问权的主体。（参阅第261页的『第28章 使用访问控制表』：以获取 ACL 描述。）例如，假设文件 X 的 ACL 包含两入口: 一个允许组 A 写访问，另一个允许组 B 读访问。然后，同为组 A 和 B 成员的主体可读写文件 X。

项目列表和权限

主体只从与其登录的名称或别名关联的组中产生项目列表访问权。它们不从其名称和所有别名中产生权限。例如，假设名为 **gustav** 的主体同是组 A 和 B 的成员。在别名 **gus**、**gustav** 下也同是组 C 和 D 的成员。主体作为 **gustav** 登录时，主体只从组 A 和 B 中产生访问权。主体使用别名 **gus** 登录时主体只从组 C 和 D 中产生访问权。

要显示拥有主体（或其别名）的组，使用 **principal show** 命令，它在第345页的『第34章 查看注册表信息』中描述。

禁止项目列表上的包含

如果从项目列表的包含中禁止组，则不产生其权限。例如，再次假设文件 X 的 ACL 包含两入口：一个允许组 A 读访问 X 文件，另一个允许组 B 写访问 X 文件。假设将项目列表包含特性设置为不允许在项目列表中包含组 B。同为组 A 和 B 成员的主体要访问文件 X，只允许他读访问而不允许写访问。如果项目列表包含特性允许组B 存在于项目列表中，则同为组 A 和 B 的成员将获得读写访问权限。

可以决定从该列表的包含中禁止某些组。例如可以从任何项目列表的包含中禁止带有类似 root 访问权的任何保留组。

添加组和组织

使用 **dcecp group create** 命令添加组，使用 **dcecp organization create** 命令添加组织。当添加组或组织时，必须指定组或组织的主名称。另外，可以提供表21 中列出的属性选项。

注意，使用 **dcecp group create** 命令和 **dcecp organization create** 命令时，可以使用一个命令用相同方法创建多个组或组织，这与创建多个主体相似。参阅第285页的『添加主体』 以获取详细信息。

表 21. 创建组和组织的属性选项

信息	含义
-gid	与组或组织关联所需的 UNIX 标识。可显式地输入该号或允许自动生成它。输入的数不能超过使用 **dcecp registry modify** 命令设置的最大允许 UNIX 号（**maxuid** 属性）；但是可输入一个小于使用 **registry modify** 命令为组和主体设置的小 UNIX 号（**minuid** 属性）。如果允许自动指派该号，它将在由小 UNIX 号和最大 UNIX 号定义的范围内。
-fullname *string*	用于更完整地描述主名称的可选名称。 要包含空格，将全名用花括号括起来。缺省值为空。
-inprojlist *值*	只用于组，确定组能否在项目列表上。缺省值是 **yes**。

添加组

下列示例显示如何将名为 **symphonists** 的组添加到**注册表**:

```
dcecp> group create symphonists
dcecp>
```

在示例中，自动生成组 UNIX 标识，不提供全名，该组包含在项目列表中。

添加组织

下列示例显示了如何将名为 **classic** 的组织添加到注册表:

```
dcecp> organization create classic
dcecp>
```

在示例中，自动生成组织 UNIX 标识，不提供全名。

更改组和组织

对于组和组织，可更改主名称和全名。另外，对于组可更改该组是否出现在项目列表中，对于组织可更改策略。参阅第353页的『第35章 维护策略和特性』：以获取关于更改组织策略的详细信息。）

使用 **dcecp group modify** 命令修改更改组。下列示例显示了使用带有 **-inprojlist** 选项的该命令将组 **symphonist** 的项目列表包含特性从 **yes**（包含在项目列表上）更改为 **no**（在项目列表上禁止）。

```
dcecp> group modify symphonists -inprojlist
no
dcecp>
```

使用 **dcecp group rename** 命令更改组的主名称或使用 **dcecp organization rename** 命令更改组织的主名称。这些命令具有下列格式:

```
group
rename old_name -to new_name
organization rename old_name -to new_name
```

其中:

old_name
是要更改的组或组织的主名称。

new_name
是组或组织的新的主名称。

下列示例显示了使用 **group rename** 命令将主名称从 **symphonists** 更改为 **symphonists7**:

```
dcecp> group rename symphonists -to symphonists7
dcecp>
```

注意，如果更改主名称，这种更改将在所有组和组织的成员资格列表中反映出来，组或组织作为成员在该列表中列出。

注: 该 DCE 发行版不支持将聚集的主体、组或组织 (pgo) 移动成它自身的子代。例如，不允许下列注册表操作序列:

```
dcecp -c group create foo/bar
dcecp -c group create foo
dcecp -c group rename foo -to foo/xyz
```

删除组和组织

如果删除组或组织，同时将自动删除使用该组或组织的任何帐户。例如，如果删除组 **symphonists**，将同时删除帐户 **vivaldi symphonists baroque** 和 **mozart symphonists classic**。

使用 **dcecp group delete** 命令删除组，使用 **dcecp organization delete** 命令删除组织。下列示例显示了用来删除组 **symphonists** 的 **group delete** 命令:

```
dcecp> group delete symphonists
dcecp>
```

下一个示例显示了用 **organization delete** 命令删除组织 **classic**:

```
dcecp> organization
delete classic
dcecp>
```

注意，可以通过使用 **group delete** 或 **organization delete** 命令将要删除的名称放入花括号并用空格隔开（正如您删除多个主体一样）删除多个组和组织。

维护会员资格列表

每个组或组织都拥有成员资格列表，它列出了是组或组织成员的主体。组成员可能是本地或外部单元中的主体。组织的成员必须只来自本地单元。使用 **dcecp group add** 命令向成员资格列表添加成员，使用 **dcecp group remove** 命令从列表除去成员。

如果从组或组织删除成员，将自动删除与该组或组织关联的已删除成员的任何帐户。例如，如果从组 **symphonists** 删除主体**mahler**，将同时删除帐户 **mahler symphonists classic**。

注意从组或组织删除主体将影响主体的对象权限。该更改仅在主体刷新的票券认可的票券时生效。参阅第311页的『 第31章 创建和维护帐户 』 以获取票券刷新的更多信息。

成员资格列表上帐户创建的作用

创建帐户时，创建帐户的主体必须是在帐户中命名的组或组织的成员。例如，如果创建帐户 **mahler symphonists classic**，则主体 **mahler** 必须是 **symphonists** 组和 **classic** 组织的成员。

添加与删除组成员

下列示例显示了使用带有 **-member** 选项的 **dcecp group add** 命令将 **mahler** 添加到组 **symphonists** 并将 **strauss** 从组 **symphonists** 中删除:

```
dcecp> group add symphonists -member mahler
dcecp> group remove symphonists -member strauss
dcecp>
```

注意，可以通过提供主体的全限定名将外部单元成员添加到本地组中。组织成员必须只能来自本地单元。

可使用一个 **group add** 或 **group remove** 命令添加或除去多个成员。要完成该操作，将成员名放入引号中并用空格分隔。例如，要将主体 **bach**、**britten** 和 **mahler** 添加到组 **symphonists**，请输入下列命令:

```
dcecp> group add symphonists -member
{bach britten mahler}
dcecp>
```

特殊情况下，当已将主机登录到 DCE 单元时正在更改主机的组名称信息，则现有的主机凭证将变为无效，除非执行附加步骤将主机凭证更新为新的组名称信息。

secval 进程管理主机凭证，它在 DCE 主机上执行安全性客户机功能。通常，在主机刚启动后， **secval** 进程将主机凭录到 DCE 单元，获取主机凭证并将它们存储在主机上。释放并重新激活 **secval** 进程以在更改组名称信息后更新这些凭证。下列示例说明了远程主机 **persephone** 上的这些操作:

```
dcecp> secval deactivate /.:/hosts/persephone/config/secval
dcecp> secval activate /.:/hosts/persephone/config/secval
dcecp>
```

创建和维护主体或组的别名

使用 **dcecp principal create** 命令创建并维护主体和组的别名。不能将别名给予组织。

创建别名

要创建主体的别名，请输入下列格式的 **dcecp principal create** 命令:

principal create *name* **-uid** *unix_ID* **-alias yes**

其中:

name 是主体或组的别名。

unix_ID
是与正在为其创建别名的主体关联的 UNIX 标识。

-alias 表示 *name* 是别名。

要创建组的别名，请输入下列格式的 **dcecp group create** 命令。

group create *name* **-gid** *group_ID* **-alias**
yes

其中:

name 是主体或组的别名。

group_ID
是与正在为其创建别名的组关联的 UNIX 标识。

-alias 表示 *name* 是别名。

将主名称更改为别名及将别名更改为主名称

注: 有关 “DCE 安全性注册表” 和 “LDAP 集成” 特性的主题信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

要将别名更改为主名称或将主名称更改为别名，对于主体使用 **dcecp principal modify** 命令或对组使用 **dcecp group modify** 命令。这些命令具有下列格式:

principal modify *name* **-alias {yes|no}**
group modify *name* **-alias {yes|no}**

其中:

name 是要更改为别名的主名称，或是要更改为主名称的别名。

-alias 指定 **-alias yes** 将由 *name* 标识的主名称更改为别名；指定 **-alias no** 将由 *name* 标识的别名更改为主名称。

主体或组只能同时具有一个主名称。将别名更改为主名称之前，首先应将主名称更改为别名。

第31章 创建和维护帐户

所有主体都有两个标识: 提供访问整个网络中机器上 DCE 对象的能力的网络标识，和提供访问本地机器上对象的能力的本地标识。两个标识同时存在，但是相互独立。主体的网络标识由帐户在网络注册表中定义。主体的本地标识由本地数据（如存储在本地机器上的 **/etc/passwd** 和 **/etc/group** 文件中的项）定义。如果使用了 **passwd_export** 命令来用存储在本地注册表中的数据更新 **/etc/passwd** 和 **/etc/group** 文件，则从存储在网络注册表中的信息派生本地标识数据。

注册表帐户通过将主体与组、组织和相关的帐户信息（如用来认证主体标识的口令）关联来定义网络标识。必须为每个进行网络通信的主体（不管通信是否是经过认证的）主体创建一个注册表帐户。必须为之创建注册表帐户的主体有如下几种:

- 访问网络中对象的每个实际用户；除非特别将用户限制在本地机器，否则这可能包括所有实际用户。
- 访问网络中的对象、并在其自己的标识（而不是在启动它的主体的标识）下运行的每一台服务器。
- 网络中的每一台机器。
- 参与单元间认证通信的单元。（单元间认证的帐户是特殊类型的帐户，将在第339页的『第33章 管理多单元环境』描述。

本章描述:

- 每种类型的帐户以及如何创建和维护它
- 如何认证帐户以及如何显示特权属性和票券
- 如何创建和维护存储服务器主体密钥的密钥表文件
- 如何维护本地注册表

用户帐户

用户帐户与用户登录到 DCE 时所用的口令和信息相关。帐户信息包括以下内容，如主体主目录和登录外壳，以及定义协助控制主体对 DCE 访问权的参数的认证策略。使用 **dcecp account create** 命令来创建实际用户的帐户，使用 **dcecp account modify** 命令来修改它们，使用 **dcecp account delete** 命令来删除它们。

服务器帐户

参与网络中通信的服务器也称为*应用程序*，它们可以运行在其自己的网络标识或启动它们的网络标识下。要在其自己的标识下运行，必须将服务器编程，以使服务器执行登录并认证那个标识。因此，必须使用 **dcecp account create** 命令来创建这些服务器的注册表帐户。

服务器帐户的口令

在登录期间，所有主体（人、服务器和机器）都必须将它们的口令传递给 DCE 认证服务器，该服务器使用这些口令来生成认证密钥。真人用户的最常用方法是简单地输入其口令。而对于服务器主体，则需要提供不同的方法。建议的方法是与 DCE 一起提供的基于 API 的方法，该方法用来将服务器密钥存储在本地受保护的密钥表中。 DCE 提

供的 API 的缺省实现将密钥表存储在服务器本地机器上的密钥表文件中，并保护该文件，以便只有主体的本地标识可以读或写该文件。

您可以远程访问密钥表文件。在本地机器上，将密钥表文件存储在不被任何文件系统导出的机器磁盘的某个分区中。

除了以 root 运行或运行在本地机器的标识下的服务器之外，需要对每台服务器使用单独的密钥表文件。在登录期间，服务器可以访问该文件以获得其密钥、将其密钥传递给认证服务、登录并认证。

使用 **dcecp keytab add** 命令将服务器密钥添加到密钥表文件，并使用 **dcecp keytab remove** 命令删除服务器密钥。

创建服务器帐户的步骤

要创建服务器的帐户，首先运行 **dcecp account create** 命令创建帐户，然后运行 **dcecp keytab add** 命令向密钥表文件中添加一项。注册表中的服务器口令和密钥表文件中的服务器密钥必须匹配。可以通过在两个命令中手工输入同样的口令来确保这些口令是相同的，或者可以指定: **keytab add** 命令应该在它设置密钥表文件中的服务器口令时重置服务器的注册表口令。

机器帐户

所有机器必须在注册表中也有帐户。机器帐户，与服务器帐户一样，通过首先运行 **account create** 命令创建帐户，然后运行 **keytab add** 命令将服务器口令添加到密钥表文件中来创建。与服务器帐户一样，注册表中与密钥表文件中的机器帐户口令必须匹配。机器帐户中的主体名称必须与单元名称空间的机器名称相同。（请参阅 *IBM DCE 版本 3.2 AIX 和 Solaris 版: 管理指南--简介* ，以获得有关单元名称空间中名称的更多信息。）

如何认证由帐户表示的标识

当主体登录到 DCE 时，安全性客户机使用它们提供的口令（或者如果是服务器或机器主体，则为这些主体提供的口令）来派生主体的认证密钥。在注册表数据库中还存在主体认证密钥的副本，该副本是在创建主体的帐户时（或者更改口令时）存贮在那里的。因此，认证服务可以使用它。

认证服务使用这个密钥来认证主体（也就是说，确认主体的身份），如下所示:

1. 安全性客户机执行以下步骤:
   a. 查询用户的口令，并使用该口令来派生主体的认证密钥
   b. 准备登录请求，部分请求用认证密钥认证
   c. 将请求转发到认证服务
2. 认证服务执行以下步骤:
   a. 除去登录请求
   b. 获得主体认证密钥的注册表副本
   c. 试图用这个密钥解密登录请求

如果解密成功，则密钥相同，主体因此被认证，登录也成功。

如果解密失败，则由主体提供、并由安全性客户机使用来派生其主体的认证密钥版本的口令无效（也就是说，与用来派生主体认证密钥的注册表副本的口令不同），登录被拒绝.

这里只大体上给出了一些介绍，有关主体的详细讨论，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Application Development Guide--Core Components*。

特权属性

认证完主体之后，DCE 安全性服务帮助获得主体的特权属性。特权属性由代表主体网络标识的 UUID、主体所处的组和与该主体相关的扩展属性组成。当主体请求访问对象以确定它们对那些对象拥有的权利时使用这些属性。 DCE 安全性服务所提供的特权属性是经过认证的。网络服务接受经过认证的特权。未经认证的特权可能不会被接受。这意味着: 根据主体的特权属性是否被认证，允许主体访问 DCE 对象的访问权类型可能不同。（在第261页的『第28章 使用访问控制表』中描述了基于主体的特权属性来控制对 DCE 对象的访问的 DCE ACL。）

授予票券的票券和对于服务的票券

授予票券的票券允许主体请求和接收对于 DCE 服务的票券（如对于 “分布式文件系统” 服务器的票券），以读取文件。允许主体访问 DCE 服务的票券被称为*服务票券*。

授予票券的票券和服务票券都有由个人帐户的设置以及注册表策略和特性所决定的生存期。当主体的授予票券的票券失效时，主体不再被认为是经过认证的用户。未经认证的主体对本地机器上对象之外的其它对象的访问权大为减少，而且主体使用 DCE 服务的能力也变得极受限制。要改正这点，必须通过运行 **kinit** 命令来重新认证主体（请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference*）或者先注销然后再重新登录 DCE。

主体的服务票券由认证服务自动延期，无需主体方进行任何操作。清注意，分配给服务票券的生存期永远不得超过主体的*授予票券的票券* (TGT) 上剩余的时间。

显示特权属性和票券

DCE 单元管理员可以使用 **klist** 命令来显示主体的当前票券和特权属性。 **klist** 命令显示三种类型的信息: 特权属性、失效信息和服务票券信息。 DCE 用户还可以运行 **klist** 来显示其当前和已失效的票券。*IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference* 中描述了 **klist** 命令。

klist 显示的第一部分--特权属性

klist 命令显示主体的特权属性。该显示首先列出全限定主体名称，然后是单元的 UUID 和名称、主体名称（不带单元名称空间和 DCE 全局标识）和该主体所处的所有组。该部分的 **klist** 显示样本如下:

```
DCE 标识信息:
   全局主体: /.../dresden.com/music/mozart
   单元: 5ad96550-80c4-11ca-b26c-08001e039431 /.../dresden.com
   主体: 00000066-80c5-11ca-b600-08001e039431 music/mozart
   组: 00000003-80c4-11ca-b201-08001e039431 composers
```

klist 显示的第二部分--失效日期和时间

klist 显示的第二部分显示主体的授予票券的票券、帐户和口令失效的日期和时间:

- 第一行显示授予票券的票券失效的日期和时间. 在这发生之前，主体应该通过运行 **kinit** 或重新登录到 DCE 来重新初始化它。
- 第二行显示主体的帐户何时失效。如果帐户失效，则主体无法登录到 DCE。要改正这点，DCE 管理员必须更改注册表中主体的帐户失效日期。
- 第三行显示主体口令失效的日期。在这发生之前，主体应该通过使用 **dcecp** 更改口令。如果口令失效，则主体无法登录到 DCE。要改正这点，DCE 管理员必须更改注册表中主体的口令。

klist 显示的第二部分样本如下:

```
标识信息失效: 91/10/03:12:07:18
帐户失效: 91/12/31:12:00:00
口令失效: 91/10/31:12:00:00
```

klist 显示的第三部分--票券

klist 显示的第三部分，也是最后部分，显示主体的票券信息和主体的票券高速缓存名称。以下显示中，头三个标号为 **Server** 的票券是在主体登录并获得特权属性之后所使用的票券。所有主体的显示都有这些项。

其余标号为 **Client** 的票券显示了主体的授予票券的票券和服务票券。在字 **Client** 之后的每个票券的列表中，显示了特权服务器的名称，即是在 DCE 安全性服务器认证了主体的身份之后授予特权属性的服务器。主体拥有的票券所适用的服务器名称在 **Server** 项之后显示，这些票券的有效日期和时间在下行显示。例如，在下面的样本显示中，最后一行显示: 主体拥有到名为 **file_server** 的服务器的票券。这个票券的生存期是从 1991 年 10 月 2 日下午 1:24 零 2 秒到 1991 年 10 月 3 日下午 12:07 零 18 秒。（时间以 24 小时的格式显示。）

```
Kerberos 票券信息:
票券高速缓存: /tmp/dcecred_17a80000
缺省主体: music/mozart@dresden.com
服务器: krbtgt/dresden@dresden.com
   有效 91/10/02:12:07:18 到 91/10/03:12:07:18
服务器: dce/rgy@dresden.com
   有效 91/10/02:12:07:20 到 91/10/03:12:07:18
服务器: dce/ptgt@dresden.com
   有效 91/10/02:12:07:49 到 91/10/03:12:07:18
客户机: dce/ptgt@dresden       服务器: krbtgt/dresden@dresden.com
   有效 91/10/02:12:07:50 到 91/10/03:12:07:18
客户机: dce/ptgt@dresden.com   服务器: dce/rgy@dresden.com
   有效 91/10/02:12:07:53 到 91/10/03:12:07:18
客户机: dce/ptgt@dresden.com   服务器: file_server@dresden.com
   有效 91/10/02:13:24:02 到 91/10/03:12:07:18
```

破坏主体的票券

使用 **kdestroy** 命令来使主体已经获得的票券无效。当主体注销时，主体的票券没有被破坏，它们保持有效，直到失效为止。 DCE 用户想在他们注销之前使用 **kdestroy** ，以便没有有效票券保留。然而，如果主体在内核中有票券高速缓存，则当主体的最后一个进程终止时破坏主体的票券。这意味着，通常没有必要在注销时运行 **kdestroy**。

IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference 中描述了 **kdestroy** 命令。

添加帐户

使用 **dcecp account create** 命令将帐户添加到注册表。与帐户相关的信息大体上分为以下两个类别:

- 与通常可以在 **/etc/passwd** 文件中发现的信息类似的用户信息。
- 可让您控制帐户的网络访问权的认证策略。认证策略建立帐户和口令有效性、帐户失效策略和票券失效策略。对认证策略控制得越严格，您的单元就越安全，但是因而增加的系统开销也越多。

这两种类型的信息都作为标准 **dcecp** 属性列表中的属性或作为属性选项提供。

请注意，也可以为注册表设置认证策略。如果注册表策略与为帐户输入的策略不同，则应用更严格的策略。(有关相互矛盾的策略的详细信息，清参阅第353页的『第35章 维护策略和特性』。)

表22 列出了创建帐户所用的属性选项。请注意，这个表中所描述的选项也可以在属性列表中不带破折号提供。

表 22. 创建帐户所用的属性选项

选项	含义
-acctvalid {yes|no}	确定帐户有效性的标志。如果将这个标志设置成 **no**，则帐户无效，并且帐户主体无法登录到帐户。缺省值是 **yes**。
-client {yes|no}	指出帐户是否用于可作为客户机的主体的标志。如果将该帐户设置成 **yes**，则主体可以登录到该帐户并获得认证的票券。缺省值是 **yes**。
-description *string*	可移植字符集 (PCS) 格式的文本字符串，通常用来描述帐户的使用。无缺省值。
-dupkey {yes|no}	确定向帐户的主体发出的票券是否可以有多个密钥的标志。缺省值是 **no**。
-expdate	帐户失效的日期(ISO 时间戳记格式 *YY-MM-DD-hh:mm:ss*)。要在帐户失效之后延期帐户，更改日期。缺省值是 **none**，表示帐户永不失效。
-forwardabletkt {yes|no}	一个标志，确定是否可以向帐户的主体发出具有与当前 TGT 网络地址不同的网络地址的新的授予票券的票券。(**-proxiabletkt** 属性对服务票券执行相同的功能。)缺省值是 **yes**。
-goodsince *date*	已知帐户最后处于未泄漏状态的日期和时间(ISO 时间戳记格式 *YY-MM-DD-hh:mm:ss*)。任何在这个日期之前授予的票券都无效。如果知道帐户的口令被泄漏，那么对这个日期的控制特别有效。更改口令可以防止未经授权的主体使用那个口令重新访问系统，但是如果在更改口令之前骗取了系统组件的票券，则不能防止该主体访问该系统组件。要消去主体对系统的访问权，必须取消票券。将 **-goodsince** 属性设置成更改泄漏口令的日期和时间，以使在那个时间和之前发出的所有票券无效，并消去未经授权主体的系统访问权。当创建帐户时，**-goodsince** 属性被设置成当前日期。

表 22. 创建帐户所用的属性选项 (续)

选项	含义
-group *group_name*	与帐户相关联的组名称。必须提供这个属性以创建帐户，无缺省值。
-home *dir_name*	登录时放置主体的目录。无缺省值。
-organization *org_name*	与帐户相关联的组织。必须提供这个属性以创建帐户，无缺省值。
-password *password*	帐户所需的明文口令。系统对您提供的口令进行加密。无缺省值。
-postdatedtkt {yes|no}	一个标志，确定是否可以向帐户的主体发出具有未来开始时间的票券。缺省值是 **no**。
-proxiabletkt {yes|no}	一个标志，确定是否可以向帐户的主体发出具有与当前票券不同的网络地址的新票券。（**-forwardabletkt** 属性选项对授予票券的票券执行相同的功能。）缺省值是 **no**。
-pwdvalid {yes|no}	确定当前口令是否有效的标志。如果将这个标志设置成 **no**，则帐户口令已经失效，并且在主体下次登录到帐户时将提示主体更改口令。缺省值是 **yes**。
-renewabletkt {yes|no}	DCE 目前不使用 Kerberos V5 可延期票券特性，目前不支持可延期票券属性的任何使用。
-server {yes|no}	指出帐户是否用于可作为服务器的主体的标志。如果帐户用于参与认证通信的服务器，则将这个标志设置成 **yes**。缺省值是 **yes**。
-shell *path_to_shell*	当主体登录时执行的外壳。
-stdtgtauth {yes|no}	确定向帐户的主体发出的票券是否可以使用授予票券的票券认证机制的标志。缺省值是 **yes**。
-usertouser {yes|no}	对于服务器，确定服务器是否必须使用用户到用户认证的标志。值或者是 **yes**（必须使用基于用户到用户协议的认证）或 **no**（使用基于服务器密钥票券协议的认证）。缺省值是 **no**。
-maxtktlife *hours*	最大票券生存期。这是票券保持有效的最大时间量（以小时计）。当客户机请求对于服务器的票券时，授予该票券的生存期将服务器和客户机的 **maxtktlife** 属性值都考虑在内。换句话说，不能超出服务器或客户机最大票券生存期的最小值。 如果不指定帐户的 **maxtktlifetime** 属性值，则使用为注册表授权策略定义的 **maxtktlifetime** 属性值。（请参阅第 35 章。）

表 22. 创建帐户所用的属性选项 (续)

选项	含义
-maxtktrenew *hours*	最大票券可延期周期。这是在主体的授予票券的票券失效、那个主体必须重新登录到系统来重新认证并获得另一个授予票券的票券之前的时间量（以小时计）。 主体的服务票券的生存期永远不能超出该主体的授予票券的票券的生存期。将证书的最大可延期时间制定得越短，系统的安全性就越好。然而，由于主体必须重新登录以延期其授予票券的票券，这个时间应该考虑到用户是否方便和所需的安全性级别。 如果不指定用户的 **maxtktrenew** 属性值，则使用为注册表授权策略定义的 **maxtktrenew** 属性值。（请参阅第 35 章。）当刷新服务票券时，DCE RPC 当时不使用可延期的票券功能。然而,它受 “DCE 安全性服务器” 的支持，并且对把 “DCE 安全性服务器” 用作 KDC 的 Kerberos V5 应用程序是很有用的。

注: 可以用 **dcecp registry modify** 命令将最大票券生存期和最大票券可延期时间当作整体来设置注册表的注册表特性。当用 **dcecp account create** 或 **account modify** 命令设置它们时，它们仅应用到某一个特定的帐户。

设置票券生存期

应该知道其它两个由 **dcecp registry modify** 命令设置的选项，即，缺省票券生存期和最小票券生存期:

- 最小票券生存期--可以分配给票券的最短生存期。请注意，最小票券生存期的实际有效值受缺省证书生存期的影响。
- 缺省票券生存期--除非主体特别请求不同的生存期，否则就是授予票券的生存期。尽管主体可以请求票券的特定生存期，但是大多数都接受缺省生存期。（如果主体请求的生存期为 0（零），则为票券分配缺省生存期。）

  请注意，缺省票券生存期的实际有效值受最大证书生存期的影响。

分配给票券的实际生存期取决于由 DCE 安全性服务所实施的最大票券生存期、缺省票券生存期和最小票券生存期设置方面的规则。这些规则如下:

- 最大票券生存期永远不能大于可延期票券的生存期（换句话说，**max_life = min (max_life, renewable_life)**）并小于 60 秒。如果最大票券生存期大于可延期的票券生存期，则使用可延期的票券生存期作为最大票券生存期。例如，假定将某个帐户设置成 15 小时。如果将可延期的票券的生存期设置成 20 小时，则有效的票券生存期不是 20，而是 15 小时。
- 缺省票券生存期永远不能大于最大票券生存期（换句话说，**default_life = min (default_life, max_life)**）并小于 60 秒。如果缺省票券生存期大于最大票券生存期，则使用最大票券生存期作为缺省票券生存期。例如，假定注册表策略指定了 25 小时的缺省票券生存期。如果将注册表的最大票券生存期设置成 15 小时，则注册表的有效缺省证书生存期不是 25，而是 15 小时。
- 最小票券生存期永远不能大于缺省证书生存期（换句话说，**min_life = min (min_life, default_life)**）并小于 60 秒。如果最小票券生存期大于缺省证书生存期，则使用缺省

票券生存期作为最小票券生存期。例如，假定注册表策略指定了 10 小时的缺省票券生存期。如果将帐户的最小票券生存期设置成 15 小时，则帐户的有效最小票券生存期不是 15，而是 10 小时。

尽管 **dcecp** 允许您输入与规则相反的值，并在您查看帐户的策略（用 **account show** 命令)时显示这些值，但是使用的值是在规则中描述的值，而不是输入的值。

注: 要做到精确，必须将网络中的时钟与和注册表数据相关的时间同步。

授予票券的票券生存期和服务票券生存期

认证服务永远不会为主体授予生存期超过主体的授予票券的票券生存期中所剩时间的服务票券。例如，如果主体的授予票券的票券的生存期只剩下 2 小时，而主体请求和接受了 4 小时的缺省服务票券生存期，则只授予 2 小时的生存期。

如果设置了主体帐户的可延期票券标志（**renewabletkt** 属性），则该主体的授予票券的票券的生存期也影响服务票券的延期。服务票券延期的时间不能超过主体授予票券的票券所剩的生存期。通常为服务票券的生存期延期即是为初始票券分配的生存期。如果初始时间超过了授予票券的票券的生存期，则将票券只延期授予票券的票券所剩的时间。

添加帐户示例

使用 **dcecp account create** 命令来创建帐户。当使用 **account create** 命令时，必须提供与所创建帐户对应的主体和与帐户相关的组和组织。另外，必须用 **-mypwd** 选项提供您的口令以验证您的身份。如果不输入您的口令，**dcecp** 将提示您。允许所有其它属性取缺省值。请注意，如果提示您输入口令，则输入的口令不会显示在屏幕上。

因为需要您输入口令，所以，必须在非交互模式运行 **account create** 命令。您不能以命令行模式运行它，那样，您的口令将显示在屏幕上。

下例演示了如何使用 **dcecp account create** 命令创建与组 **symphonists** 和组织 **classic** 相关的主体**mahler** 的帐户。允许所有其它帐户属性取缺省值。

```
dcecp> account
create mahler -group symphonists -organization classic \
> -password passwd -mypwd cellpwd
dcecp>
```

请注意，可以用一个 **account create** 命令创建多个帐户。要这样做，将与正在创建的帐户所对应的主体用空格分隔括在花括号中。例如，要创建主体 **bach**、**britten** 和 **mahler** 的帐户，可能会输入以下命令:

```
dcecp> account create
{bach britten mahler} -group symphonists \
> -organization classic -password music -mypwd cellpwd
dcecp>
```

当创建多个帐户时，为每个帐户分配相同的属性。这意味着，在上例中，**bach**、**britten** 和 **mahler** 的帐户都与 **symphonists** 组和 **classic** 组织相关，并且它们都被分配了口令 **music**。您可能会发现，用这种方法为属于同一组和组织的主体创建多个帐户是很有用。要通知那些用这种方法创建的帐户所对应的用户立即更改其口令。

修改帐户

dcecp account modify 命令允许您修改帐户。可以修改任何帐户属性。

当修改帐户时，必须用 **-mypwd** 选项提供您的口令，以验证您的身份。请注意，如果提示您输入口令，则输入的口令不会显示在屏幕上。如果不输入口令，将提示您输入。因为需要您输入口令，所以，必须在非交互模式运行 **account modify** 命令。您不能以命令行模式运行它，那样，您的口令将显示在屏幕上。

当使用 dcecp **account modify** 命令设置用户帐户的 **maxtktrenew** 或 **maxtktlife** 属性值时，您可能会接收到错误

```
msgID=0x17122084 无效数据记录。
```

这可能会在第一次设置这两个帐户属性的任意一个时发生，即使为这些属性指定了有效值时也是如此。要第一次设置帐户上的 **maxtktrenew** 或 **maxtktlife** 属性，必须在 dcecp **account modify** 命令上指定这两个属性。为帐户设置了两个属性之后，可以使用 use dcecp 来分别修改它们中的任意一个。

下例演示了如何使用 **account modify** 命令指定 **mahler** 的帐户的新的主目录:

```
dcecp> account modify mahler -home /.../music/fs/users/mahler/concert
\
> -mypwd cellpwd
dcecp>
```

请注意，可以在 **account modify** 中使用 **-change** 选项，以在属性列表中提供更改。**account modify** 命令不支持 **-add** 和 **-remove**选项，因为每个帐户属性都必须存在并且必须有值。

删除帐户

下例演示如何用 **dcecp account delete** 命令删除主体 **mahler** 的帐户:

```
dcecp> account delete  mahler
dcecp>
```

如果删除组或组织，则也将自动删除与那个组或组织相关的任何帐户。

可以用一个 **account delete** 命令删除多个帐户。要这样做，将帐户名称用空格分隔括在花括号中。例如，要删除 **bach**、**britten** 和 **mahler** 的帐户，可能输入:

```
dcecp> account
delete {bach britten mahler}
dcecp>
```

创建、维护和删除密钥表文件

以下 **dcecp** 命令允许您创建、维护和删除密钥表文件:

keytab create
: 创建密钥表文件及其所有密钥项。

keytab delete
: 删除密钥表文件及其所有密钥项。

keytab add
: 将密钥项添加到密钥表文件。

keytab remove
从密钥表文件除去密钥项。

以下小节描述如何管理密钥表文件。

密钥表文件

密钥表文件存储在与它们所包含密钥的服务器所在的同一台机器上。可以使用 **dcecp** 来远程或本地访问它们。对于远程访问，**dcecp** 使用 **dced** 接口。 **dcecp keytab** 命令的 **-local** 选项允许您不使用 **dced** 而访问本地密钥表文件。

因为 **dced** 提供对密钥表文件的远程访问，所以文件被定义成 **dced** 对象，并且那些对象被存储在 **keytab** 目录下由 **dced** 控制的名称空间中。 **dced** 密钥表文件由标识对象的 UUID、可选的注释和在本地机器上实际存储服务器密钥的文件名称组成。这个对象通常是个文件。

请注意，实际的服务器密钥不存储在密钥表对象中，而是存贮在本地机器上的文件中。

dced 密钥表对象的路径名是:

/.:/hosts/*hostname*/**config/keytab/***keytab_name*

其中:

hostname
是 **dced** 进程所驻留主机的名称。

keytab_name
是密钥表文件的名称。

本地密钥表文件的路径名是:

/opt/*dcelocal*/*keytab_path_name*

其中:

keytab_path_name
是本地机器上密钥表文件的路径名。

保护密钥表文件

必须对本地密钥表文件加以足够的保护，使它们在网络上不可用。因为它们在缺省 DCE 实现中使用，所以，密钥表文件包含主体密钥，这些密钥是 DCE 安全性的基础。如果这些密钥被泄漏，则网络的安全性也会被泄漏。访问密钥表文件的调用使用 **rpc_c_protect_level_pkt_privacy**。这种保护级别在传递的数据上执行数据加密标准 (DES) 的加密。 **dcecp keytab -noprivacy** 选项允许您指定应该使用您站点的缺省保护级别。

为在每个本地节点上运行的每一台服务器主体创建一个单独的密钥表文件。共享同一密钥表文件的服务器可以访问彼此的密钥，因而可以相互伪装。 保护密钥表文件，以便只有 root 用户可以访问它们。如果不这样做，则必须由 root 用户启动服务器，以便在登录期间读他们的密钥表文件并获得其密钥。

当创建或更改服务器密钥时，可以为在本地节点上运行的每一台服务器命名不同的密钥表文件。保护该文件，以便只有密钥在该文件中包含的服务器可以访问它。然后将服务器文件的 **setuid** 位设置成服务器的标识，以便服务器可以访问该密钥表文件并获得其密钥。

服务器和机器密钥版本号

当将密钥添加到密钥表文件中时，每一个密钥被分配一个范围从 1 到 255 的版本号。每当服务器或机器的密钥更改时（自动地或显式地），密钥的版本号就增加。版本号允许任何给定的服务器或机器有两个或更多个密钥存在。当密钥更改时，任何仍然使用在未更改的旧版本密钥下被授予的票券的服务器或机器将不中断而继续运行，直到票券自然失效为止。当票券失效时，服务器或机器被重新认证并获得新的密钥。

如果对 **keytab add** 命令使用 **-registry** 选项，则如果可能的话，将自动删除旧密钥。如果不使用该选项，则应当偶尔使用 **keytab list** 命令来列出密钥表文件的内容，并使用 **keytab delete** 命令来删除过时的旧版本。

注: 从密钥表文件删除密钥时要当心。当更改主体密钥时，基于您所删除密钥的票券可以继续存在。如果从密钥表文件删除密钥，服务器将不接受任何基于被删除密钥的活动票券，并且拥有那些票券的客户机将出现认证故障。

创建和维护密钥和密钥表文件

两个命令允许您创建密钥项:

keytab create
创建密钥表文件、密钥表文件项和 **dced** 密钥表对象。

keytab add
将密钥项添加到现有的密钥表文件。

当运行这两个命令时，需提供要创建和修改的密钥表文件名。

表23 列出了可以应用到 **keytab create** 和 **add** 命令的其它选项。

表 23. keytab create 和 keytab add 选项

选项	含义
-local	不使用 **dced** 来访问密钥表文件。
-entry	只创建 **dced** 配置信息，而不创建实际密钥表。
-noprivacy	指定使用的保护级别应该是您站点的缺省保护级别，而不是 **rpc_c_protect_level_pkt_privacy**。
-member *name*	其密钥正在被创建和更改的主体（服务器或机器）名称。可以在一个列表中提供多个名称。如果提供了一个列表，则为在该列表中命名的所有主体分配同一密钥。
-key *key*	提供给帐户的明文文本。该选项不能与 **-random** 选项一起使用。
-random	生成随机密钥。如果使用该选项，则必须使用 **-registry** 选项来将远程生成的密钥添加到注册表中服务器或机器的帐户。该选项不能与 **-key** 选项一起使用。

表 23. keytab create 和 keytab add 选项 (续)

选项	含义
-registry	更新注册表中的主体密钥，使其与您输入（或自动生成）的密钥表文件中的密钥匹配。使用它来确保：当更改密钥表文件中主体的密钥时，注册表中主体和密钥表文件中的密钥同步。 当使用 **-random** 选项时需要该选项。使用这个选项可能需要运行 **dcecp login** 命令，来确保您的网络标识适合于修改注册表数据库。如果用户要使用 **--registry** 选项，则主体的口令（密钥）必须在密钥表文件中存在。
-version *number*	指定密钥的版本号。如果不使用 **-registry** 选项，则必须使用该选项。
-storage *local_file_name*	要创建的本地文件名。这个选项只对 **keytab create** 命令使用。当向现有的密钥表文件添加项时，用 **dced** 对象名标识文件。
-data *keys*	服务器主体名称和密钥，格式为: *principal_name key_type* { *version*} {*key_value*}

创建密钥表文件

使用 **keytab create** 命令创建密钥表文件、文件中的项和相应的 **dced** 对象。当使用这个命令时，必须提供要创建的 **dced** 对象的路径名作为自变量、指定密钥表本地位置的 **storage** 选项、指定服务器主体和密钥名称的 **data** 选项，以及在第321页的表23 列出的任何合适的选项。

该 **data** 选项的形式为:

principal_name key_type {*version*} {*key_value*}

其中:

principal_name
是其密钥表文件正在创建的服务器主体名称。

key_type
是指定以明文文本还是 DCE 加密格式存储密钥的代码:

- **des** 表示 DES 加密。
- **plain** 表示明文文本。

version 是密钥的版本号。如果不提供版本号，则为密钥分配版本号 **1**。

key_value
除非指定 **-random** 选项来随机生成密钥，否则必须提供一个密钥值。如果 *key_type* 是 **plain**，则必须提供明文文本格式的密钥。如果 *key_type* 是 **des**，则必须提供一个 DCE 加密的密钥。

以下样本命令执行这些任务:

1. 创建 **dced** 密钥表对象 **/.:/hosts/music/config/keytab/svr4_key**
2. 在名为 **music** 的本地机器上的 **keys** 目录中创建名为 **/opt/dcelocal/keys/svr4_key** 的密钥表文件。
3. 在文件中创建主体 **mahler** 的明文文本密钥项，并为其分配版本号 **3**。

```
dcecp> keytab create
/.:/hosts/music/config/keytab/svr4_key -attr \
> {{storage /opt/dcelocal/keys/svr4_key} \
> {data {mahler plain 3 mon#Repos}}}
dcecp>
```

在密钥表文件中添加项

使用 **keytab add** 命令向现有的密钥表文件中添加项。当您使用这个命令时，必须提供密钥表文件的 **dced** 对象名称和第321页的表23 中所描述的任何选项。

以下命令将一个密钥添加到服务器主体 **svr_3** 的名为 **kfile_3** 的密钥表文件中。将自动生成密钥，并会更新注册表，以与密钥表文件同步。

```
dcecp> keytab add /.:/hosts/foo/config/keytab/kfile_3 \
> -member svr_3 -random -registry
dcecp>
```

从密钥表文件除去项

可以使用 **dcecp keytab remove** 命令从密钥表文件除去项。当使用这个命令时，必须提供密钥表文件的 **dced** 对象名。

当使用 **keytab remove** 命令时，必须提供密钥表文件的名称和密钥将被删除的主体名称（或主体列表）。

还可以提供 **-version** 选项来指定要删除的密钥版本号，提供 **-type** 选项来指定要删除的密钥（表示明文文本的 **plain** 或者表示 DES 加密密钥的 **des**）。如果使用 **-version** 或 **-type** 选项，则将只删除指定版本或类型的密钥。

以下命令除去密钥表文件 **/.:/hosts/foo/config/keytab/kfile_3** 中主体 **svr_2** 的所有 DES 密钥。

```
dcecp> keytab remove /.:/hosts/foo/config/keytab/kfile_3 \
> -members svr_2 -type des
dcecp>
```

除去密钥表文件

可以通过使用 **dcecp keytab delete** 命令来除去本地和密钥表文件及其相关的 **dced** 对象。

要删除本地密钥表文件和 **dced** 对象，需为命令提供本地文件名。可以用一条命令将多个密钥表文件用空格分隔并括在花括号中来删除多个密钥表文件。例如，以下命令删除密钥表文件以及 **dced** 对象 **/.:/hosts/foo/config/keytab/kfile_2** 和 **/.:/hosts/foo/config/keytab/kfile_3**。

```
dcecp> keytab delete {/.:/hosts/foo/config/keytab/kfile_2 \
> /.:/hosts/foo/config/keytab/kfile_3}
dcecp>
```

要只删除 **dced** 对象，使用 **-entry** 选项。

例如，下列命令除去名为 **/.:/hosts/foo/config/keytab/kfile_3** 的 **dced** 对象，但是保留本地文件 **/opt/***dcelocal***/keys/kfile_3** 不动。

```
dcecp> keytab delete -entry /.:/hosts/foo/config/keytab/kfile_3
dcecp>
```

更改密钥表文件中的服务器和机器口令

当所有主体的口令失效时必须更改这些口令。真人主体可以使用他们平台的 **chpass** 命令来更改他们的口令。 **dced** 安全性确认服务通过分配一个随机生成的口令来在需要时自动更改机器的口令。这个守护程序与 DCE 一起提供，并在参与网络访问的每一台本地机器上运行。一般地，您可以假设：其它供应商创建的服务器或应用程序也通过随机生成口令来在需要时自动更改其口令。然而，如果运行在其自己标识下的服务器不自动更新其口令，则您必须使用 **dcecp keytab add** 命令手工执行，如第323页的『在密钥表文件中添加项』所述。

注: 运行在真人主体标识下的服务器不应该自动更新其口令。当这样的服务器更新其口令时，它也更新它所运行下的真人主体的口令。然后，真人主体必须提供这个随机生成的口令以登录到系统并重新认证。因为真人主体永远不知道随机生成的口令，所以主体无法登录到系统，也无法重新认证。

处理密钥表文件中泄漏的服务器或机器口令

如果泄漏了服务器或机器的口令，必须执行以下步骤来在注册表和服务器本地密钥表文件中更改它:

1. 使用 **keytab remove** 命令删除泄露的口令。
2. 使用 **keytab add** 命令创建服务器或机器的新口令。
3. 如果不使用 **keytab add** 命令的 **registry** 选项来与服务器或机器的密钥表文件同时更新服务器或机器的注册表帐户，则运行 **account modify** 命令来更改注册表中的服务器或机器口令，以便与密钥表文件中的口令匹配。

维护本地注册表

如果不能使用网络注册表，则本地注册表允许从那台机器登录。如果网络服务器在运行，则在真人和非真人用户第一次从本地机器登录到 DCE 时自动创建本地注册表。当用户登录到机器时，他们的帐户信息自动添加到本地注册表。

以下文件构成本地注册表数据库:

*dcelocal***/var/security/lrgy_data**
包含帐户信息项。

*dcelocal***/var/security/lrgy_tgts**
包含授予票券的票券项。

*dcelocal***/var/security/lrgy_lock**
由安全性服务器使用来锁定注册表，以进行读 / 写操作。

必须使用安全性命令 **rgy_edit** 维护本地注册表。可以使用以下子命令:

view 查看本地注册表中的所有项。

delete *principal_name*
从注册表删除由 *principal_name* 指定的主体。

properties
设置和查看本地注册表的特性。

help 获得有关本地注册表的帮助信息。

除了前面列表中的子命令之外，**rgy_edit** 命令还提供一个附加的子命令 **purge** 和一个附加的、名为*注册表容量*的注册表特性，来限制本地注册表中的项数量。

本节以下部分描述 **purge** 子命令和注册表容量特性。

注册表容量特性

rgy_edit properties 命令可让您设置本地注册可以包含的项数量限制。例如，假设将注册表容量限制为 **10** 并且注册表包含 10 个项。如果新用户从本地机器登录，则为那个用户创建一项。这个新项覆盖初始 10 项中的最老一项。当用户从机器进行第一次登录时，他们新创建的注册表项覆盖最老的项。

设置容量和生命期特性

可以设置本地注册表可以包含的项数量限制。这个限制由本地注册表对象的 **Capacity** 属性设置。将这个属性的值设置成适当的项数量可以防止本地注册表过大增长。

例如，假设将机器上本地注册表的 **Capacity** 属性值设置成 **10**，并且本地注册表包含 10 项。如果新用户从本地机器登录，则为那个用户创建一项。这个新项覆盖当前在本地注册表中存在的 10 项中最老的一项。当用户从机器进行第一次登录时，他们新创建的注册表项覆盖最老的项。

要设置本地注册表的 **Capacity** 属性，使用 **rgy_edt properties** 命令。

请注意，当第一次输入该命令时，它显示当前注册表容量和生命期。另外，它显示本地注册表中帐户项的数量和 TGT 项的数量。有一个包含每个帐户项的帐户凭证的 TGT 项。

```
$ rgy_edit -l
rgy_edit=> properties
本地注册表特性:
  容量: 25 项
  包含: 6 个帐户项
  包含: 6 tgt 项
  生命期: 3w
要作更改吗 [y/n]? (n) y
输入本地注册表容量: (25) 50
输入 acct 生命期的天数或 'forever': (3w) 12w
```

在前面的示例中，**12w** 指定了 12 周的生命期。当输入生命期时，可以输入:

- **forever**--指定项永远不会失效。然而，如果它超过了注册表容量，将覆盖项。在**输入 acct 生命期的天数或 'forever': (3w)**提示处输入 **0**（零）与输入 **forever** 的效果相同。
- 十进制数--指定测量单位（星期、天、小时、分钟、秒）的数量。这个数字通常与测量单位一起使用，如下所描述。如果只输入数量而不输入测量单位，则测量单位缺省为天。
- **w**、**d**、**h**、**m** 或 **s**--指定测量单位:
  - **w**，星期
  - **d**，天
  - **h**，小时
  - **m**，分钟
  - **s**，秒

可以使用测量单位及其伴随数量的任意组合。例如，要将生命期设置成 12 星期、8 小时和 30 秒，则输入以下命令:

```
输入 acct 生命期的天数或 'forever': (3w) 12w8h30s
```

如果一个数量和测量单位的字符串以单独一个数量结束，则不带测量单位的数量缺省为秒。例如，如果输入以下内容，则假定生命期为 12 个星期 30 秒:

```
输入 acct 生命期的天数或 'forever': (3w) 12w30
```

清除失效的项

rgy_edit purge 子命令从本地注册表删除失效的项。当运行 **rgy_edt purge** 命令时，该命令删除所有生命期属性已指明失效的项。

在前一个示例中，我们将本地注册表生命期的属性设置成 **12w**。将生命期设置成 **12w** 之后，下面的 **rgy_edt purge** 命令删除所有存在时间超过 12 个星期的本地注册表项（帐户和 TGT）。

```
rgy_edit=> purge
清除了 1 个帐户
清除了 1 个 TGT 项
```

第32章 创建与使用扩展注册表属性

注册表中存储了有关主体、组、组织和帐户的特定信息。这些信息是在使用 **dcecp** 命令创建主题、组、组织和帐户时创建的。可以存储在注册表数据库中的此类信息是以注册表模式定义的，它基本上是存储在数据库中的此类数据的目录。这是一个可以与注册表对象关联或附加到注册表对象的每个属性类的模式项定义。例如，模式项定义主体名称作为 DCE PCS 格式的可打印字符串。在创建主体时，可以输入 PCS 格式的文本字符串。

可以使用扩展注册表属性 (ERA) 设施来添加定义所选属性类型的模式项。这些属性称为*扩展属性*，因为它们扩展了注册模式。一旦定义了扩展属性类型，就可以使用 **dcecp create** 或 **modify** 操作把它们添加到安全性对象。所创建的扩展属性类型被应用于和 DCE 一起运行的定制应用程序中，并被传递到那些应用程序进行处理。例如，如果执行一个需要用户的 MVS™ 名称的应用程序 MVS，则可以建立一个存储在注册表中的 MVS 名称扩展属性。然后，这个 MVS 名称就可以传递到 MVS 应用程序进行适当的处理。

如果主体具有扩展属性，则这些属性将和认证主体时获得的扩展特权属性 (EPAC) 被执行。

在本手册中，属性类型与定义了扩展属性类型的模式项有关。“属性”实例与附加到注册表对象并具有值的属性有关。

本章描述如何创建和维护属性类型和属性实例。首先将讨论 **xattrschema** 对象；然后描述如何为对象定义属性类型和附加属性。

xattrschema 对象

扩展属性类型存储在 CDS 名称空间中联结点（通常为 **/.:/sec**）下名为 **xattrschema** 的对象。对 **xattrschema** 以及它所包含的属性类型定义的访问由 **xattrschema** 对象上的 ACL 控制。与其它注册表类似，**xattrschema** 对象也是从主安全性服务器传送到副本的。

创建和维护属性类型

使用 **dcecp xattrschema** 命令来创建和维护属性类型。作为命令的一部分，必须提供属性类型的全限定名称（例如 **/.:/sec/xattrschema/***name*）作为变量。

创建属性类型

使用 **dcecp xattrschema create** 命令来创建属性类型。以下将概述可以提供给该命令的属性选项。注意，此表中描述的选项使用在属性列表中时也可以不加破折号。

用于创建扩展属性的选项

选项　含义

-aclmgr *description*

ACL 管理器类型的必需列表，它支持该属性类型可以附加到的对象以及那些管理器支持的许可权。无缺省值。属性类型 ACL 管理器将在第330页的『定义属性的 ACL 管理器』中完整地描述。

-annotation *string*

注释属性类型的 PCS 文本字符串。如果字符串中包含空格，则用花括号或引号括起。缺省值为空白。

-applydefs

此选项当前无法实现。

-encoding *type*

属性类型实例值的格式。第331页的『定义属性类型编码』中将更详细地描述属性编码。

-intercell

此选项当前无法实现。

-multivalued {yes | no}

表示该属性是多值还是非多值（**yes**= 多值；**no**= 非多值）。如果属性是多值的，则具有相同属性类型的多个实例可以附加到单一注册表对象。例如，属性 A 被编码为多值的，则单一主体将可以有多个属性 A 的实例。如果它被编码为非多值的，则单一主体只能有一个属性 A 的实例。缺省值为 **no**。

-reserved {yes | no}

表示属性是否可保留（**yes**= 可保留；**no**= 不可保留）。可保留属性类型不能被删除，除非除去了保留的限制。缺省值为 **no**。

-scope *name*

当前发行版中无法实现。

-trigtype *type*

标识了触发器服务器是否与属性类型关联的，如果关联，则还将标识触发器的类型。可能的值如下：**none**--触发器服务器与属性类型无关。**query**--查询触发器服务器与属性类型关联。**update**--更新触发器服务器与属性类型关联。如果 **-trigtype** 选项被设置为 **query** 或 **update**，则必须提供 **-trigbind** 选项来指定触发器服务器的绑定。更详细的信息，请参阅第332页的『定义属性触发器服务器』。

-trigbind *binding*

如果触发器服务器与属性类型关联，则此选项要指定触发器的服务绑定。

-unique {yes | no}

表示属性类型的每个实例在单元内是否必须是唯一的（**yes**= 唯一的；**no**=不是唯一的）。例如，假设属性类型 A 的一个实例与单元内的 25 个主体相连。如果属性类型 A 被编码为唯一的，则这 25 个主体的每个 A 属性的值必须各不相同。如果没有编码为唯一，则这 25 个主体的每个 A 属性的值可以是相同的。缺省值为 **no**。

-uuid *uuid*

UUID 标识了内部的属性类型。注意，作为变量提供给 **dcecp xattrschema create** 命令的名称是用于访问属性类型的。如果不提供 UUID，则系统将生成一个。

dcecp xattrschema create 命令语法如下:

```
xattrschema create attr_name{attr_options}
```

其中:

attr_name
是要创建的属性类型的全限定名。

attr_option
是上述列表中描述的一个或多个选项。

下面这个样本命令将创建名为 **employee_num** 的扩展属性，并为该属性指派一个 ACL 管理器 **principal** 以及编码类型 **integer**:

```
dcecp> xattrschema create /.:/sec/xattrschema/employee_num \
> -aclmgr {principal r r r r} -encoding integer
dcecp>
```

虽然这个样本使用选项来提供信息，但是您也可以使用标准的 **dcecp** 属性列表。

注意，您可以使用一个操作来提供一个名称列表以创建多个模式项。但是，您应当知道，如果命令变量中包含一个以上的模式名，则不能指定 UUID 属性，并且您所指定的属性将应用于所创建的所有项。

修改属性类型

使用带 **-change** 选项的 **dcecp modify** 命令来修改属性类型。只有 **aclmgr**、**applydefs**、 **intercell**、**trigbind**、 **annotation** 和 **reserved** 模式类型属性可以修改。

xattrschema modify 命令语法如下:

xattrschema modify *attr_name* **-change** *new_option*

其中:

attr_name
要更改的属性类型的全限定名。

new_option
指定更改的选项。

以下样本命令修改了 **MVSname** 属性，以更改其注释。注意，必须向命令提供全限定属性类型名。

```
dcecp> xattrschema modify
/.:/sec/xattrschema/MVSname -change \
> {annotation {Use with version 3.0}}
dcecp>
```

重命名属性类型

使用 **dcecp xattrschema rename** 命令来更改扩展属性的名称。输入如下格式的命令:

xattrschema rename *old_name* **-to** *new_name*

其中:

old_name
要更改的全限定扩展属性名。

new_name
新的简单扩展属性名。

以下示例显示了 **xattrschema rename** 命令，它将把属性名从 **log_name** 更改为 **MVSname**。

```
dcecp> xattrschema rename /.:/sec/xattrschema/log_name -to MVSname
dcecp>
```

删除属性类型

使用 **dcecp xattrschema delete** 命令删除扩展属性。请注意，在删除一个属性类型的同时该属性类型的所有实例也将被删除。例如，假设 **MVSname** 属性的一个实例附加到了名为 **delores** 的主体。如果删除 **MVSname** 属性，则附加到 **delores** 的属性的实例也将被删除。

要删除属性类型，请输入如下格式的命令:

xattrschema delete *attribute_name*

其中 *attribute_name* 是要删除的属性的全限定名。

例如，要删除名为 **MVSname** 的扩展属性，则该命令将是:

```
dcecp> xattrschema delete /.:/xattrschema/MVSname
dcecp>
```

定义属性的 ACL 管理器

当定义扩展属性类型时，必须定义该属性可以附加到的对象以及用于访问该属性的许可权。要做到这点，应把属性与一个或多个 ACL 管理器关联起来，并提供控制访问该类型属性实例的许可权设置。该属性只能附加到 ACL 管理器集所命名的 ACL 管理器类型支持的对象。并且，只有在 ACL 管理器集中命名的许可权才适用于访问属性实例。（注意，这些许可权不同于那些控制对象 ACL 管理器并已经由它们建立的那些许可权。）例如，假设名为 **MVSname** 的属性类型的 ACL 管理器集仅列出了主体的 ACL 管理器类型。那么，名为 **MVSname** 的属性类型的实例只能附加到主体而不能附加到其它任何注册表对象。**MVSname** 属性的 ACL 管理器集还包含了控制对 **MVSname** 属性访问的许可权。

使用 **dcecp xattrschema -aclmgr** 选项指定属性的 ACL 管理器集。这个选项的格式如下:

{*mgr_uuid queryset updateset testset deleteset*}

其中:

mgr_uuid

表示 UUID，标识了将与属性类型关联的 ACL 管理器。您既可以提供 UUID，也可以提供以下速记名中的一个（它们将在内部转换为 UUID）来访问由 DCE 提供的 ACL 管理器类型:

policy 用于访问策略对象的 ACL 管理器。

principal
用于访问主体的 ACL 管理器。

group 用于访问组的 ACL 管理器。

organization
用于访问组织的 ACL 管理器。

secdirectory
用于访问注册表数据库中目录的 ACL 管理器。

replist 用于访问副本列表的 ACL 管理器。

xattrschema
用于访问注册表模式的 ACL 管理器。

srvrconf
用于访问 **dced** 对象的 ACL 管理器。

queryset
用于查询属性实例的许可权设置。

updateset
用于修改属性实例的许可权设置。

testset 用于测试属性实例的许可权设置。

deleteset
用于删除属性实例的许可权设置。

要输入具有一个以上许可权的许可权设置，请并置这些许可权；例如对于要输入 **t**、**M** 和 **d**，可输入 **tMd**。

每个 ACL 管理器的信息都用花括号括起，并且每一项之间都保留一个空格（当然，并置的许可权设置除外）。

例如，可以用以下命令为 **MVS_name** 属性定义一个附加的“ACL 管理器”：

```
dcecp> xattrschema
modify /.:/sec/xattrschema/MVS_name \
> -aclmgr {18dbdad2-23df-11cd-82d4-080009251352 r w t mD}
dcecp>
```

该命令为 **MVS_name** 属性添加了一个由 UUID **18dbdad2-23df-11cd-82d4-08000925135** 标识的 ACL 管理器。ACL 管理器的许可权设置如下:

- **r** 表示查询许可权设置。
- **w** 表示更新许可权设置。
- **t** 表示测试许可权设置。
- **mD** 表示删除许可权设置。

注意，不能修改或删除属性类型的 ACL 管理器集。但是，可以向它添加附加的管理器类型。

定义属性类型编码

必须定义在属性类型编码中可以为属性类型提供的值的格式。只有那些具有编码中定义的格式的值才能赋值给属性。例如，编码可以指定属性类型的实例中包含的值只能是 UUID 格式的。

每个属性类型只能有一种编码，并且不能修改该编码。此外，特殊编码类型使您能创建属性设置。

使用 **dcecp xattrschema -encoding** 选项来指定属性的编码。这个选项具有以下格式:

-encoding *type*

参数 *type* 表示下面所要描述的编码类型中的一个。

编码类型

编码类型

含义

any 这个属性值可采用任何编码。此编码类型只在定义模式项中的 ERA 时才有效。ERA 的所有实例都必须具有一种使用其它一些值的编码。

attrset

这个属性值必须是一个用花括号括起的属性类型 UUID 的列表。这种编码类型定义了一个属性集。属性集更易于搜索和检索。例如，对某一属性集的查询将返回属于该属性集中的属性的所有实例。

binding

这个属性值必须包含适合与 DCE 服务器通信的认证、权限以及绑定信息。如果要把一个名称或服务器绑定作为对象的属性存储，请使用此编码。详细信息，请参阅第334页的『指定绑定信息』。

byte 这个属性值必须是字节字符串。字节字符串被假设为保留类型或一个自定义类型。注意，这种编码类型允许输入二进制数据。有关输入二进制数据的更详细信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference*。

confidential

在 DCE 的本发行版中无法实现。

i18ndata

这个属性值必须是一个国际化的字节字符串，具有一个标识用于编码数据的 OSF 注册代码集的标记。注意，这种编码类型允许输入字节数据。有关输入字节数据的更详细信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference*。

integer

该属性必须是一个带符号的 32 位整数。

printstring

该属性实例值必须是一个 PCS 可打印的值。

stringarray

这个属性值必须是多个打印字符串中的一个数组。注意，打印字符串可以为空。

这个属性值必须是多个打印字符串中的一个数组。注意，打印字符串可以为空。

这个属性值必须是 DCE UUID。

void 这个属性没有值。它只是一个或显示或消失的标记。

定义属性触发器服务器

当属性值存储在外部数据库时，有些属性类型需要有外部服务器的支持来验证输入属性值或提供输出属性值。例如，这类服务器可以将旧的注册表系统与 DCE 注册表相连接。属性触发器设施提供了对外部 DCE 服务器的自动调用，称为*属性触发器*。

当要查询或更新与触发器服务器关联的某个属性时，将自动调用触发器服务器。注意，访问触发器服务器所维护的信息完全由该服务器来控制。

注: 本发行版不支持更新触发器服务器。

要把一个属性类型与触发器服务器关联起来，请使用 **-trigtype** 和 **-trigbind dcecp xattrschema** 选项。

-trigtype 选项

-trigtype 选项定义属性类型是否与触发器服务器关联，如果关联，那么是什么类型的服务器。这个选项的格式如下:

```
-trigtype [none | query | update]
```

其中:

none 表示属性与触发器服务器无关联。(这是缺省值。)

query 表示属性与查询触发器关联。查询触发器服务器只能执行查询。

update
表示属性与更新触发器关联。更新触发器可以执行查询与更新。

注: 本发行版中不支持更新触发器服务器。

一旦设置了 **-trigtype** 选项就不能修改。

-trigbind 选项

-trigbind 选项定义了有关触发器服务器以及触发器自我绑定的认证信息。

如果 **-trigbind** 选项具有以下格式。

```
-trigbind {{auth_info}{binding_info}}
```

下列部分描述如何指定认证类型和绑定。

指定认证类型

auth_info 参数有以下语法:

```
{auth_serv_type name prot_level authentication_service authorization_service}
```

其中:

auth_serv_type
指定了认证类型，它们可以是

- **none**--不执行认证。
- **dce**--执行标准的 DCE 认证。

如果不使用认证，则除自我绑定之外不需要其它任何信息。如果使用标准的 DCE 认证类型，则必须指定其余的所有参数。

name 指定触发器服务器的主体名称。

prot_level
指定保护级别，确定由认证服务保护的客户机和服务器之间的受认证通信的程度。可能的保护级别有:

- **default**--使用缺省的保护级别 **pkt**。

- **none**--不执行认证：不交换凭证，不建立会话密钥，不认证客户机 EPAC 或名称并且清除传输。注意，虽然不能信任未认证的 EPAC，但是它们在调试、跟踪和探测时可能有用。
- **connect**--只有在客户机与服务器建立关系时才认证。
- **call**--只有当服务器接收到请求时，才在每个远程过程调用的开始处认证。

  这个级别不适用于从基于连接的协议序列（也就是 **ncacn_ip_tcp**）上发出的远程过程调用。如果指定为这一级别并且绑定句柄使用基于连接的协议序列，则通常使用 **pkt** 保护级别来代替它。
- **pkt**--确保接收的所有数据都来自期望的客户机。
- **pktinteg**--确保并验证未修改任何客户机与服务器之间的数据传送。这是最高级别的保护级别，受到 RPC 运行时的保证。
- **pktprivacy**--由前面所有的级别指定的认证并且还加密每个 RPC 变量值。这是最高级别的保护级别，但不受 RPC 运行时的保证。

authentication_service

指定认证服务。认证服务提供的保护的确切级别是由保护级别指定的。支持的认证服务如下:

- **default**-- 共享 DCE 密钥。
- **none**--无认证：不交换凭证，不建立会话密钥，不传输客户机 EPAC 或名称并且清除传输。指定为 **none**，将关闭对使用该绑定发出的远程过程调用的认证。
- **secret**--共享 DCE 密钥认证。

authorization_service

指定授权服务。授权数据的有效性和可信性，如任何的应用程序数据，依赖于所指定的认证服务和保护级别。支持的授权服务如下:

- **none**--服务器不执行授权。只有当授权服务设置为 **none** 并指定不执行认证时才有效。
- **name**--服务器根据客户机主体名称执行授权。如果授权服务为 **none**，则不能使用这个值。
- **dce**--服务器使用通过用这个绑定产生的每一个远程过程调用发送给服务器的客户机 DCE EPAC 来进行认证。通常，访问时要检查 DCE ACL。

指定绑定信息

binding_info 参数指定了绑定，它可以是一个字符串绑定、服务器入口名称或一个包含一个或多个字符串绑定或服务器入口名称的列表。下列示例显示了服务器入口的名称绑定:

```
./.:/hosts/host_name/dce_entity_name
```

下列示例显示了标准语法格式的字符串绑定:

```
ncadg_udp_ip:130.105.96.3[1234]
```

下列示例显示了 TCL 语法格式的字符串绑定:

```
ncarn_ip_tcp
10-29.58.00 2001
```

-trigbind 选项的样本值

下列样本显示了 **-trigbind** 选项的值。在这个样本中，绑定的主体名称是 **MVS_server**，它使用 packet-privacy 保护级别的认证、使用共享密钥并使用 DCE 授权服务。绑定作为服务器入口名称提供。

```
-trigbind {{dce MVS_server pktprivacy secret dce} \
    {/.:/hosts/host_name/dce_entity_name}}
```

创建和维护属性实例

使用 **dcecp**，可以将扩展注册表属性附加到对象，修改指派给那些属性的值以及如同向对象添加任何属性一样删除附加的属性。

可以使用 **dcecp create** 和 **modify** 将扩展注册表属性附加到以下的任何对象:

- **principal**
- **group**
- **organization**
- **policy**

  注: 在 DCE 版本 1.2.1 中，不能向策略对象附加属性。
- **directory**
- **replist**
- **xattrschema**

向对象附加属性实例

当使用 **dcecp principal -attribute** 操作创建对象时，可以将属性附加到对象，或者，您可以使用 **dcecp modify -add** 操作将属性附加到现有的对象中。

例如，要创建主体 **delores** 并在同时又附加值为 **admin** 的 **MVSname** 属性，请使用下面的 **principal create** 命令:

```
dcecp> principal create delores -attribute {MVSname admin}
dcecp>
```

要添加值为 **admin** 的 **MVSname** 属性到名为 **delores** 的主体，请使用以下的 **principal modify** 命令:

```
dcecp> principal modify delores -add {MVSname admin}
dcecp>
```

要附加多值的扩展属性的实例，请在属性名后用空格包含每个值。例如，要将值为 **value1**、**value2**、**value3** 和 **value4** 的 **multi_name** 属性附加到名为 **delores** 的主体，请使用以下命令:

```
dcecp> principal modify delores -add {multi_name value1 value2 \
> value3 value4}
dcecp>
```

修改属性实例

使用 **dcecp modify -change** 操作更改属性实例的值。属性是否可修改由使用该属性的应用程序确定。例如，以下命令将把指派给名为 **delores** 的主体的 **MVSname** 的值从 **admin** 更改成 **cell_admin**。

```
dcecp> principal modify delores -change {MVSname cell_admin}
dcecp>
```

如果按上述段落所显示的 **dcecp modify -change** 命令来更改多值属性的值，则该多值属性的所有实例将被删除并用命令中指定的新值替换它们。例如，若只想更改某一指定的值，则必须输入所有的值。例如，假设 **multi_name** 属性有以下这四个值: **value1**、**value2**、**value3** 和 **value4**。若要把 **value4** 更改成 **value5**，则必须输入以下命令:

```
dcecp> principal
modify delores -change {multi_name {value1 value2 \
> value3 value5}}
dcecp>
```

但是，可以从多值属性中添加和删除个别值。使用 **-add** 选项来添加值。例如，假设 **multi_name** 属性的值有 **value1**、**value2**、**value3** 和 **value5**。以下样本命令将值 **value6** 添加到 **multi_name** 属性。

```
dcecp> principal modify
delores -add {multi_name value6}
dcecp>
```

（使用下面子节中所述的 **remove** 选项来删除多值属性中的特定值。）

注意，以下命令将用值 **value1** 替换名为 **delores** 的主体带有的属性 **multi_name**的所有实例:

```
dcecp> principal modify delores -change {multi_name value1}
dcecp>
```

例如，如果 **multi_name** 属性有以下值:

```
{multi_name value1 value2 value3}
```

那么，上述命令将把其值更改为如下:

```
{multi_name value1}
```

删除属性实例

使用带 **-remove** 选项的 **dcecp modify** 命令删除附加到对象的属性实例。要删除对象的所有属性实例，应为 **-remove** 选项提供该属性名称。例如，下列命令将删除名为 **delores** 的主体的 **MVSname** 属性的所有实例:

```
dcecp> principal modify delores -remove MVSname
dcecp>
```

要删除多值属性的单一实例，应提供属性名称和属性值。例如，以下命令将仅从名为 **multi-value** 的多值属性删除实例 **value5**。而其它值与属性本身将保持不变。

```
dcecp> principal modify delores -remove {multi-value value5}
dcecp>
```

但是，如果删除多值属性的最后一个实例，则 **dcecp** 还将从对象中删除该属性，因为如果属性没有值就不能附加在对象后。注意，不能使用同一个命令同时删除多值属性以及多值属性的值。

若要从对象中删除一个以上的属性，则必须使用 **-types** 选项。这个选项将告诉 **dcecp**，提供的所有值都是属性类型的名称，而不是属性值。例如，以下样本命令将使用 **-types** 选项来删除主体 **delores** 中名为 **MVSname** 和 **MVSinteger** 的属性:

```
dcecp> principal modify delores -remove {MVSname MVSinteger}
-types
dcecp>
```

如果不使用 **-types** 选项，那么 **dcecp** 将假设 **MVSinteger** 是 **MVSname** 属性的值，因为不存在这样的值，命令将失败。

使用属性集

属性集是属性 UUID 的一个集合，而 UUID 标识了该集合的成员——实例。属性集把与对象相关的属性实例组成一组，使它们更易于被访问。例如，如果使用 **dcecp show** 操作来显示一个属性集，则该显示将扩充属性集并在显示输出中包含集中的所有属性。这个属性扩充只能在显示信息的 **dcecp** 命令中使用。用于创建和修改属性实例的命令只能在命令中命名的指定属性中使用。因为，属性是属性集的成员，但它们的存在独立于属性集，因此可以象操作其它任何属性一样操作它们。

每个属性集都和一个对象相连，虽然系统没有强求，但属性集中每个成员属性最好应该与同一个对象相连。属性集不能嵌套；属性集中的成员，其本身不能是属性集。

要创建、修改和删除属性集中的成员，请按创建、修改和删除多值属性的指示执行。属性集中的成员属性实例由 UUID 来标识。

扩展注册表属性限制

在创建扩展注册表属性 (ERA) 模式时，只建议使用 **-unique** 属性 (**sec_attr_sch_entry_unique**)。即使在 ERA 模式中这个属性被设置成 **yes**，但它的行为将仍然与设置成 **no** 一样。DCE 代码不检查附加到对象的 ERA 值，也不强求 ERA 值必须是唯一的。

在创建 ERA 模式时，只建议使用 **-applydefs** 属性 (**sec_attr_sch_entry_use_defaults**)。即使在 ERA 模式中这个属性被设置成 **yes**，但它的行为将仍然与设置成 **no** 一样。如果 ERA 没有显式地附加到对象，DCE 代码将不为 ERA 提供缺省值。

在创建 ERA 模式时，只建议使用 **-intercell** 属性。即使这个属性被设置成 **accept** (**sec_attr_intercell_act_accept**) 或 **evaluate** (**sec_attr_intercell_act_evaluate**)，其行为将仍然与被设置成 **reject** (**sec_attr_intercell_act_reject**) 一样。当主体的 EPAC 被用于 intercell 访问时，DCE 代码将废弃主体、组或组织的所有 ERA 值。

在创建 ERA 模式时，**-encoding** 属性的值 **confidential_bytes** 不能实现。将 ERA 模式的 **-encoding** 属性设置为 **confidential_bytes** (**sec_attr_enc_confidential_bytes**) 与 **bytes** (**sec_attr_enc_bytes**) 具有相同的行为。

对 ERA 模式的属性集编码的处理不完全支持。将 ERA 模式中的 **attr_set** (**sec_attr_enc_attr_set**) 编码设置成允许使用一个唯一的 uuid 作为组成模式集的 uuid。**sec_rgy_attr_lookup_by_id( )** API 当前的行为与 **sec_rgy_lookup_no_expand( )** 相似。不支持属性集无缝扩充到其组件。

在创建 ERA 模式时，不实现 **-trigtype** 属性的 **update** 值 (**sec_attr_trig_type_update**)。对 ERA 模式的触发器的更新将使其有能力在更新 ERA 值之前检查已注册的服务器。

第33章 管理多单元环境

本指南中的前几章描述了在单一单元中执行的 DCE 管理任务。多单元环境的管理，在其中主体从外部单元访问内部单元中的对象，以及跨越不同单元主体进行交互的引起的附加任务和考虑。

事实上您有两种类型的系统管理器：一个用于本地单元管理，另一个用于多单元环境的内部单元管理。如果设置了两种类型管理器的组，则可以设置 **krbtgt** 的 ACL 包含单元主体的目录，在注册表数据库中只允许内部单元管理器组对其更新。但是请确保允许所有其它用户读访问 **krbtgt** 目录或拒绝这些用户访问内部单元。注意，如果用该方法保护 **krbtgt** 目录，请确保 **krbtgt** 目录下的所有目录具有正确的 ACL。最容易的实现方法是创建注册表后更改 **krbtgt** 目录上的“对象 ACL”和“初始创建 ACL”。

本章描述了允许主体从外部单元访问单元中对象的单元之间的信任关系，反之亦然。

信任关系

注: *OSF DCE 版本 1.2.2 代码不提供本节中讨论的传递信任关系的支持。*

要在其它单元中给出主体的显式许可权来预定授权访问单元中的对象，则必须与该单元建立信任关系。可使用 **dcecp registry connect** 命令创建两个特殊帐户：一个在单元的注册表中表示外部单元，另一个在外部单元的注册表中表示单元。建立这些帐户表示您信任外部单元的认证服务能够正确地认证外部用户，因此如果该单元中的所有用户都由外部单元的认证服务标记为已认证，则认为它们是已认证的。

一旦建立了信任关系，就可以就象对待本地单元中的主体那样控制外部主体对带有 ACL 项的特定对象的访问。信任关系还允许外部单元中的用户登录到本地单元中的帐户，反之亦然。

两种信任关系允许其它单元中的主体参加您单元中对象的授权访问。这些关系是*直接信任关系*和分层传递信任关系。本章中使用术语*传递信任关系*表示分层传递信任关系的实现。

直接信任关系

在直接信任关系中，两个单元的认证服务共享认证密钥并且彼此信任以确认它们单一单元中的主体。因此如果每个单元中的认证服务已经分别将各自的所有用户都标记为已认证，则两个单元都将考虑认证它们。共享的认证密钥是从单一口令（每个单元各一个）中派生出来的，它用于将一个单元中的所有主体认证到另一个单元中。直接信任关系仅参与两个单元。

建立信任关系

使用**注册表连接**命令建立直接信任和传递信任关系。注意，虽然可以在两个单元之间创建直接信任关系，也只可为那些通过传递信任路径连接的单元创建传递信任关系。

该命令创建两个特殊帐户：一个在您的单元的注册表中表示外部单元，另一个在外部单元的注册表中表示单元。命令同时创建帐户的主体。一旦建立了信任关系，则外部

单元中的用户可登录到本地单元中的帐户，反之亦然。可以就象对待本地单元中的主体那样控制外部主体对带有 ACL 项的特定对象的访问。

创建帐户后，**注册表连接**命令将执行两个您应该清楚的任务。首先，它自动生成一个两个帐户共享的口令。这表示登录到一个单元的用户，只要它们所在的单元具有信任关系，则将视为相同的主体并共享同一口令。其次，**注册表修改**命令生成所有主体共享的 UNIX 号。该共享的 UNIX 号帮助防止访问本地机器上的对象时本地和外部主体之间的 UNIX 号冲突。

在注册表内为了访问网络，用表示全限定名称的 UUID 标识主体，例如对于主体 **mahler** 使用 **/.../dresden.com/dce/users/mahler**。但是，本地机器上的操作系统使用 UNIX 号标识主体。因为单元之间的 UNIX 号不需要是唯一的，不同单元中的两个主体可能具有相同的 UNIX 号。因此，正在访问本地单元中文件的外部主体可能具有与本地主体相同的 UNIX 号并且本地系统将它视为本地机器上本地用户文件的所有者。

为给定单元中访问本地单元的每个主体都创建 UNIX 号将防止以上情况的发生。但是应该清楚因为所有外部用户都具有相同的 UNIX 号，正是防止它们访问本地用户文件的特性允许它们访问彼此的文件。因为相同外部单元中的每个用户都视为相同用户，同一个外部单元中的其它外部用户都可以访问本地机器上外部用户所有的每个文件。

创建信任关系

要创建对等关系，请遵循下列步骤:

1. 运行**注册表连接**命令创建单元间认证帐户（您的单元注册表中的帐户和外部单元注册表中的其它帐户）。

   注: 如果发出了 **rgy_edit cell** 或 **dcecp registry connect** 命令，但是在其中一个单元的注册表中仍然存在其它单元的 **krbtgt** 项（从以前的信任配置获取），则可能看起来该命令已成功。但是已认证的内部单元访问可能失败，因为无法同步两个 **krbtgt** 项的关键字。

   可通过完整查看每个为内部单元创建的 **krbtgt** 帐户来检测这种情况的发生。该帐户命名为 **krbtgt** / *外部名称*并且创建在每个单元注册表中以允许对其它单元的内部单元访问。如果外部单元的 **krbtgt** 帐户上的时间与最后更改时间不同，则该项可能无效。

   要从情况恢复，请删除外部单元的 **krbtgt** 帐户和主体。这必须分别在每个单元中完成。然后再次发出 **rgy_edit cell** 或 **dcecp registry connect** 命令。可使用下列 **dcecp** 命令除去 **krbtgt** 帐户帐户和主体:

   ```
   dcecp -c account delete krbtgt/
   dcecp -c principal delete krbtgt/
   ```

2. 或者也可以使用 **account modify** 命令来细调帐户的属性，它是创建帐户时由缺省值指派的。例如，帐户的失效日期（**expdate** 属性）缺省值为 **none**。可能想要输入一个日期以确保一段时间后主动地更新帐户。
3. 请确保外部单元中的系统管理员更改该帐户的 **acctvalid** 标志，它为 **yes** 以表示该帐户有效。如果其中之一或两个帐户均无效，则不发生单元之间的通信。

registry connect 命令的命令选项

使用 **registry connect** 命令时，必须提供用来建立对等关系的外部单元的全限定名称。该名称除去了完整的路径名并且带有 **krbtgt** 前缀，作为做帐户主体的主名称。例如，如果您输入单元名称 **/.../dresden.com**，主体名称则为 **krbtgt/dresden.com**。将不更改的单元名称存储为代理的完整名称。

注意，**registry connect** 使用您的本地单元名称作为本地单元的帐户主体主名称。该名称与外部单元名称一样除去了完整的路径名并且带有 **krbtgt** 前缀。

可对 **registry connect** 命令提供下列选项:

-acctvalid、**-facctvalid**

标志帐户为有效的设置。有效的本地帐户 (**-acctvalid**) 允许外部单元中的用户登录到本地单元中的节点。有效的外部帐户 (-facctvalid) 允许本地单元中的用户登录到外部单元中的节点。缺省值是对于每个选项均**无效**。

-expdate

本地和外部单元的帐户都失效，对等关系结束并阻止两个单元中主体之间再进行已认证的通信的时间和日期。要更新该帐户，请更改该字段中的日期。缺省值是 **none**。

-facct、**-facctpw**

外部单元中的系统管理员必须提供给您外部单元中的帐户名称和口令。外部帐户必须具有创建主体和帐户的许可权。需要用该帐户访问外部注册表在外部帐户注册表中创建表示您的单元的帐户。应该仅按需要限制该帐户的时效和创建限额。

-group、**-fgroup**

与本地单元 (**-group**) 和外部单元 (**-fgroup**) 中的帐户关联的组名。这些组对于帐户没有意义并且不与外部或本地单元中的任何用户关联。必须输入它们，因为注册表需要它使所有帐户与组关联。如果不存在组，则将创建它们。

-mypwd

registry connect 命令不提示您输入正在创建的帐户口令；它随机生成该口令。但是，您必须为 **-mypw** 选项提供口令以确认您的身份。

-org、**-forg**

要与本地单元 (**-org**) 和外部单元 (**-forg**) 中的帐户关联的组织名称。这些组织对于帐户没有意义并且不与外部或本地单元中的任何用户关联。必须输入它们，因为注册表需要它使所有帐户与组织关联。如果不存在组织，则将创建它们。

创建单元间认证示例

使用下列样本 **registry connect** 命令为 **/.../dresden.com** 标识的外部单元创建帐户。本地帐户与名为 **cell_group_local**组、名为 **cell_group_dres** 和 **cell_org_dres** 的组织关联。允许帐户的失效日期缺省为 **none**。

```
dcecp> registry connect /.../dresden.com -facct cell_log
-facctpw music \
> -group cell_group_local -fgroup cell_group_dres \
> -org cell_org_local -forg cell_org_dres -mypwd cell_admin
dcecp>
```

由 registry connect 命令创建的帐户

赋予通过 **registry connect** 命令创建的帐户和主体缺省的属性值，如表24 所示。所有外部主体访问您单元中的对象时，这些属性将应用到它们中。同样地，您单元中的所有主体访问外部单元中的对象时，在外部单元中为您单元创建的帐户属性将应用到它们中。

表 24. 单元间认证主体和帐户的缺省属性值

信息	含义
帐户主体名称	本地单元帐户的本地单元名称或外部单元帐户的外部单元名称（除去完整路径名并带有 **krbtgt** 前缀。
fullname	单元的路径名。
quota	设置为 **none**。该限额应用于那些使用单元间认证帐户来访问外部单元中对象的所有主体。例如，将对象创建限额更改为 10，则通过使用该帐户访问您单元的所有外部用户可以在您单元中创建的对象总数不能超过 10。并非每个外部主体都是 10。您的单元帐户在外部单元中设置的对象创建限额与您单元中主体可在外部单元注册表中创建的对象数具有相同的限制。
description、**home**、**shell**	设置为空。
server	设置成 **yes**；即该帐户是可以参加已认证通信的服务器。
client	设置为 **no**。
pwdvalid	设置为 **yes** （有效）。
acctvalid	设置为 **no** （无效）除非使用 **-acctvalid** 和 **-facctvalid** 选项。
postdatedtkt	设置为 **yes**；即可向该帐户发出带有将来启动时间的票券。
forwardabletkt	设置为 **yes**；即该帐户发出带有与现有授予票券的票券具有不同网络地址的授予票券的票券。
renewabletkt	设置为 **yes**；即可以更新帐户的票券。
proxiabletkt	设置为 **yes**；即可向该帐户发出与现有票券具有不同网络地址的票券。
dupkey	设置为 **yes**；即该帐户票券可能有重复键。
goodsince	设置成创建该帐户的日期。
maxtktlife	设置成注册表策略。
maxtktrenew	设置成注册表策略。DCE 当前不使用 **maxtktrenew** 属性；当前不支持该选项的任何使用。

修改单元间认证帐户

任何时候都可以使用标准的 **dcecp account** 操作更改用 **registry connect** 命令创建的帐户。但是应该清楚下列注意事项。

不要将帐户的 **pwdvalid** 属性设置成 **no** （无效）。对于标准帐户，将该属性设置为 **no**，这将导致在下次登录时提示用户更改口令。两个单元中的认证服务共享单元间认证帐户口令。如果更改其中一个，就破坏了同步化而使单元间的通信结束。如果要更改认证服务共享的口令，则必须再次运行 **registry connect** 命令重新创建帐户与并创建合适的同步口令。

通常，不要删除帐户和帐户的主体，除非正在阻断单元的对等关系。如果删除了其中一个帐户，则必须运行 **registry connect** 命令重新创建两个帐户并恢复对等关系。

从 OSF DCE 1.2.2 客户机进行单元间访问

OSF 1.2.2 级别的 DCE 供应商实现通常可实现访问外部单元的新限制。对于要访问外部单元的 DCE 客户机，本地注册表（名为 **krbtgt/***foreign_cellname*）中外部单元的单元间认证帐户必须将其 **acctvalid** 标志设置成 **yes**。否则，将拒绝对外部单元的访问。

虽然 DCE 3.2 AIX 和 Solaris 版是 OSF 1.2.2 级别，但它不强制执行该限制。然而 OSF DCE 1.2.2 的其它供应商实现可能强制执行它。要允许内部单元访问，单元管理员可使用下列 **dcecp** 命令将现有的单元间认证帐户的 **acctvalid** 标志更改成 **yes**。

```
dcecp -c account modify krbtgt/ -modify \ { acctvalid yes }
```

建立新的内部单元关系时，单元管理员可使用 **dcecp registry connect** 命令的 **acctvalid** 和 **facctvalid** 标志自动将单元间认证帐户的 **acctvalid** 标志设置为 yes。

第34章 查看注册表信息

使用 **dcecp**，可以显示有关下列安全性对象的信息:

- 主体
- 组
- 组织
- 帐户
- 注册表
- **xattrschema** 对象
- ACL
- 密钥表文件

下列 **dcecp** 操作提供这些显示:

- **catalog** 命令显示所有指定对象的名称。
- **list** 显示指定组或组织的成员名称或指定密钥表的成员名称。
- **show** 命令显示有关特定对象实例的信息。

本章描述如何显示可用于所有安全性对象（除注册表对象外）的操作，有关可用于注册表对象的操作，在第361页的『第36章 执行例行维护』中有描述。.

显示帐户信息

使用 **dcecp account catalog** 和 **account show** 命令来显示有关帐户信息。当使用 **account show** 命令时，必须提供帐户的主体名，以指定要显示的帐户。要提供多个主体名，可用花括号将它们括起并用空格分隔它们。

要以字母顺序显示注册表数据库中的所有帐户并且在名称前加上单元名时，可输入:

```
account catalog
```

要以字母顺序显示注册表数据库中的所有帐户并且在名称前不加单元名时，可输入:

```
account catalog -simplename
```

要为一个已命名主体的帐户显示所有属性，可输入:

```
account show principal_name
```

要为一个已命名主体的帐户显示所有策略，可输入:

```
account show principle_name -policies
```

要为一个已命名主体的帐户显示所有属性和所有策略，可输入:

```
account show principal_name -all
```

下面的例子显示不使用 **-simplename** 选项的 **account catalog** 命令:

```
dcecp> account catalog
/.../dresden.com/bach
/.../dresden.com/bin
/.../dresden.com/brahms
/.../dresden.com/britten
```

```
/.../dresden.com/cell_admin
/.../dresden.com/daemon
/.../dresden.com/dce-ptgt
/.../dresden.com/dce-rgy
/.../dresden.com/mahler
/.../dresden.com/nobody
/.../dresden.com/root
/.../dresden.com/uucp
/.../dresden.com/hosts/pmin17/cds-server
/.../dresden.com/hosts/pmin17/gda
/.../dresden.com/hosts/pmin17/self
/.../dresden.com/krbtgt/dresden.com
dcecp>
```

下面的例子显示 **account show** 命令，以显示与 **mahler** 的帐户相关的属性:

```
dcecp> account show mahler
{acctvalid yes}
{client yes}
{created /.../dresden.com/cell_admin
1994-06-15-18:31:08.000+00:00I-----}
{description {}}
{dupkey no}
{expdate 1995-06-16-00:00:00.000+00:00I-----}
{forwardabletkt yes}
{goodsince 1994-06-15-18:31:05.000+00:00I-----}
{group users}
{home /}
{lastchange /.../dresden.com/cell_admin
1994-06-16-12:21:07.000+00:00I-----}
{organization users}
{postdatedtkt no}
{proxiabletkt no}
{pwdvalid yes}
{renewabletkt yes}
{server yes}
{shell {}}
{stdtgtauth yes}
dcecp>
```

注意，如果因为注册表策略覆盖为该帐户定义的策略，而使得它不起实际作用，则该策略后跟 **effective** 标记和有效的实际值。

显示组和组织信息

使用 **dcecp group catalog**、**group show** 和 **group list** 命令显示有关组的信息，使用 **dcecp organization catalog**、**organization show** 和 **organization list** 命令显示有关组织的信息。当使用 **group list**、**group show**、**organization list** 和 **organization show** 命令时，必须提供要显示的组或组织的名称。要提供多个名称，可用花括号将它们括起并用空格分隔它们。

要以字母顺序显示注册表数据库中的所有组或组织并且在名称前加上单元名时，可输入:

```
group catalog
```

或:

```
organization catalog
```

要以字母顺序显示注册表数据库中的所有组或组织并且在名称前不加单元名时，可输入:

group catalog -simplename

或:

organization catalog -simplename

要以字母顺序显示指定组或组织的所有成员并且在名称前加上单元名时，可输入:

group list *group_name*

或:

organization list *organization_name*

要以字母顺序显示指定组或组织的所有成员并且在名称前不加单元名时，可输入:

group list *group_name* **-simplename**

或:

organization list *organization_name* **-simplename**

要显示组或组织的所有属性，可输入:

group show *group_name*

或:

organization show *organization_name*

要显示与组或组织相连的所有扩展属性实例，可输入:

group show *group_name* **-xattrs**

或:

organization show *organization_name* **-xattrs**

要显示组或组织的所有规则属性和所有扩展属性，可输入:

group show *group_name* **-all**

或:

organization show *organization_name* **-all**

下面的例子显示不使用 **-simplename** 选项的 **group catalog** 命令:

```
dcecp> group cat
/.../dresden.com/nogroup
/.../dresden.com/system
/.../dresden.com/daemon
/.../dresden.com/uucp
/.../dresden.com/bin
/.../dresden.com/kmem
/.../dresden.com/mail
/.../dresden.com/tty
/.../dresden.com/none
/.../dresden.com/tcb
/.../dresden.com/acct-admin
/.../dresden.com/subsys/dce/sec-admin
/.../dresden.com/subsys/dce/cds-admin
/.../dresden.com/subsys/dce/dts-admin
/.../dresden.com/subsys/dce/cds-server
/.../dresden.com/subsys/dce/dts-servers
/.../dresden.com/users
dcecp>
```

下面的例子显示名为 **users_temporary** 的组的属性:

```
dcecp> group show users_temporary
{alias no}
{gid 5211}
{uuid 0000145b-9362-21cd-a601-0000c08adf56}
{inprojlist no}
{fullname {temporary users}}
dcecp>
```

注意，在前面的例子中，表示 **{alias no}** 的行。它表明名称 **users_temporary** 是主体名，而不是别名。对于别名，这一行读作 **{alias yes}**。

下列 **group list** 命令显示组 **symphonists** 的成员:

```
dcecp> group list  symphonists
/.../dresden.com/bach
/.../dresden.com/britten
/.../dresden.com/mahler
dcecp>
```

显示主体信息

使用 **dcecp principal catalog** 和 **principal show** 命令来显示有关主体的信息。当使用 **principal show** 命令时，必须提供要显示的主体的名称。要提供多个主体名，可用花括号将它们括起并用空格分隔它们。

要以字母顺序显示注册表数据库中的所有主体并且显示时在名称前加上单元名，可输入:

```
principal catalog
```

要以字母顺序显示注册表数据库中的所有主体并且显示时在名称前不加单元名，可输入:

```
principal catalog -simplename
```

要为一个已命名的主体显示所有属性，可输入:

```
principal show principal_name
```

要显示与主体相连的所有扩展属性实例，可输入:

```
principal show principal_name -xattrs
```

要显示主体的所有规则属性和所有扩展属性，可输入:

```
principal show principal_name -all
```

下面的例子显示带 **-simplename** 选项的 **principal catalog**:

```
dcecp> principal catalog -simplename
bach
bin
brahms
britten
cell_admin
daemon
dce-ptgt
dce-rgy
mahler
nobody
```

```
root
uucp
cds-server
dcecp>
```

下面的例子显示 **principal show** 命令，以显示有关主体 **mahler** 的信息:

```
dcecp> principal show /.:/mahler
{fullname {Gustav Mahler}}
{uid 30014}
{uuid 0000753e-f51f-2e0e-b000-0000c08adf56}
{alias no}
{quota unlimited}
{groups {symphonists composers}}
dcecp>
```

除了 groups 行，由 **principal show** 命令列出的所有信息都是当主体添加到注册表时创建的信息。这一行列出主体隶属于哪些组。

显示 xattrschema 信息

使用 **dcecp xattrschema catalog** 和 **xattrschema show** 命令来显示有关扩展属性类型的信息。注意，要查看与主体相连的扩展属性的实例，可使用带 **-xattr** 选项的 **principal**、**group** 或 **organization show** 命令。

xattrschema catalog 命令显示在已命名模式中定义的扩展属性对象的名称。当使用这个命令时，必须指定要显示其扩展属性的模式的名称。对于注册表数据库，这个名称是 **/.:/sec/xattrschema**。您所在的位置必须向您提供模式名称。

xattrschema show 命令显示注册表模式或您所在的位置所使用模式中的已命名模式的属性。当使用这个命令时，必须指定要显示其信息的扩展属性类型的名称。要提供多个名称，可用花括号将它们括起并用空格分隔它们。

要以字母顺序显示注册表数据库中所有属性类型的名称并且显示时在名称前加上单元名，可输入:

```
xattrschema catalog /.:/sec/xattrschema
```

要以字母顺序显示注册表数据库中的所有属性类型并且显示时在名称前不加单元名，可输入:

```
xattrschema catalog /.:/sec/xattrschema -simplename
```

要显示除注册表外的模式中的属性，可用其它模式的完全指定名称替换 **/.:/sec/xattrschema**。

要显示一个已命名的扩展属性类型的属性，可输入:

```
xattrschema show attr_name
```

下面的例子列出以单元名为前缀的注册表中所有扩展属性的名称:

```
dcecp> xattrschema catalog /.:/sec/xattrschema
/.../dresden/sec/xattrschema/pre_auth_req
/.../dresden/sec/xattrschema/pwd_val_type
/.../dresden/sec/xattrschema/pwd_mgmt_binding
/.../dresden/sec/xattrschema/X500_DN
/.../dresden/sec/xattrschema/X500_DSA_Admin
/.../dresden/sec/xattrschema/disable_time_interval
```

```
/.../dresden/sec/xattrschema/max_invalid_attempts
/.../dresden/sec/xattrschema/passwd_override
/.../dresden/sec/xattrschema/test_integer
dcecp>
```

下面的例子列出名为 **test_integer** 的扩展注册表属性的属性:

```
dcecp> xattrschema show /.:/sec/xattrschema/test_integer
{aclmgr {principal {{query r} {update r} {test r} {delete r}}}}
{annotation {test_integer: encoding type integer}}
{applydefs yes}
{encoding integer}
{intercell reject}
{multivalued yes}
{reserved no}
{scope {}}
{trigbind {none {}}}
{trigtype none}
{unique no}
{uuid 5f439154-2af1-11cd-8ec3-080009353559}
dcecp>
```

显示 ACL 信息

使用 **dcecp acl show** 命令来显示一个已命名对象的 ACL 项。当使用这个命令时，必须指定要显示其 ACL 项的对象的名称。要提供多个名称，可用花括号将它们括起并用空格分隔它们。

如果这个命令无法确定对象的名称，则它将显示对象的 UUID。

要显示指定对象的 ACL 项，可输入:

acl show *object_name*

要显示 ACL 的缺省单元，可输入:

acl show *object_name* **-cell**

要显示对象支持的 ACL 管理器，可输入:

acl show *object_name* **-managers**

下面的例子显示名为 **hosts** 的对象的 ACL 项:

```
dcecp> acl show /.:/hosts
{unauthenticated r--t---}
{user cell_admin rwdtcia}
{user hosts/absolut/cds-server1 rwdtcia}
{user root rwdtcia}
{group subsys/dce/cds-admin rwdtcia}
{group subsys/dce/cds-server rwdtcia}
{any_other r--t---}
dcecp>
```

显示密钥表信息

使用 **dcecp keytab catalog**、**keytab list** 和 **keytab show** 命令来显示有关帐户的信息。当使用 **keytab catalog** 命令时，必须提供要显示其密钥表文件的主机的名称。当使用 **keytab list** 或 **keytab show** 命令时，必须提供要显示其密钥表信息的 **dced** 对象的名称。要为任意一个命令提供多个名称，可用花括号将它们括起并用空格分隔它们。

要显示指定主机上所有密钥表文件的名称并且显示时在名称前加上单元名，可输入:

keytab catalog *host_name*

如果不提供 *host_name*，则显示列出当前主机上的密钥表文件。

要显示指定主机上所有密钥表文件的名称并且显示时在名称前不加上单元名，可输入:

keytab
catalog *host_name* **-simplename**

要显示所有主体的列表，在指定密钥表文件中有这些主体的项可输入:

keytab list *file_name*

要显示所有主体，在命名的密钥表文件中有这些主体的项可输入:

keytab show *file_name*

显示的信息只包括主体名。

要显示指定密钥文件的本地名，可输入:

keytab show *dced_object_name* **-entry**

要显示密钥文件中的所有项（包括密钥），可输入:

keytab show *dced_object_name* **-members**

下面的例子显示名为 **svr_3** 的密钥表文件中的项:

```
dcecp> keytab show /.:/hosts/music/config/keytab/svr_3 -members
{brahms des 1}
{britten plain 3}
{mahler des 2}
dcecp>
```

第35章 维护策略和特性

注册表策略是可以在整个注册表范围内注册的属性。要提供更好的控制级别，策略还可以为个别组织和帐户设置。如果组织或帐户的策略更具有限制性，则组织或帐户的策略可以覆盖注册表缺省策略。

注册表特性是适用于在注册表中创建的主体、组和组织的属性。它们不可以为个别组织或帐户设置。特性管制如可用于 UNIX 标识的数字范围以及是否显示加密口令等事情。

可以用 **dcecp registry modify** 命令同时设置策略和特性。另外，可以用 **dcecp organization modify** 和 **dcecp account modify** 命令设置个别组织或帐户的策略。在所有命令中，要设置的策略和特性都作为带 **-change** 选项的标准 **dcecp** 属性列表中的属性或属性选项提供。

本章先描述策略，然后描述特性。

策略

可以为下列各项设置策略:

- 用 **dcecp registry modify** 命令设置的整个注册表。这样，策略适用于所有主体、组和组织，除非为特定组织或帐户设置更严格的策略。
- 用 **dcecp organization modify** 命令设置的特定组织。
- 用 **dcecp account modify** 命令设置的特定帐户。

有两种策略：标准策略和认证策略。

标准策略

标准策略管制如帐户和口令寿命和口令格式等事情。它可以为注册表和特定组织设置。在以下部分中描述了您可设置的标准策略。

注: 除了本部分描述的定义口令策略外，通过将 ERA 与主体相连，可以在口令格式、口令生成、无效登录处理和失效口令处理等方面施加附加控制。有关详细信息，请参阅第283页的『第30章 创建和维护主体、组和组织』。

帐户使用期限

您设置的帐户使用期限确定帐户可用于特定组织或整个注册表的周期。过了这个时间周期，帐户失效，并且必须重新创建它。

以下面的形式，将帐户使用期限定义为 **dcecp acctlife** 属性:

acctlife {*time* | **unlimited**}

其中，*time* 是一个数字，它表示帐户的有效天数，**unlimited** 指定无限使用期限。

帐户的使用期限还受帐户失效日期（**expdate** 属性）控制，这个失效日期是您在使用 **dcecp account create** 或 **account modify** 命令来创建或更改帐户时设置的。如果

设置的帐户失效日期与帐户使用期限策略冲突，则应用更为严格的设置。例如，如果将标准策略帐户使用期限设置为 40 天，而将帐户失效日期设置为次日，则帐户在次日失效，因为它是更为严格的设置。

注: 通过使用 **dcecp account modify** 命令来将帐户标记为无效（**acctvalid** 属性），可以更及时地控制帐户的有效性。

口令使用期限

口令使用期限指定在特定组织或整个注册表的帐户口令失效前的时间周期。

通常，DCE 安全性禁止口令已失效的用户登录。但是，为了防止单元管理员因口令失效而被锁在系统之外，对于如 **cell_admin** 用户可以覆盖这个策略。通过将 *passwd_override* ERA 的实例与主体相连来完成它。有关如何完成这一操作的信息，请参阅第283页的『第30章 创建和维护主体、组和组织』。

以下面的形式，将口令使用期限定义为 **dcecp pwdlife** 属性:

pwdlife {*time* | **unlimited**}

其中，*time* 是一个数字，它表示口令的有效天数，**unlimited** 指定无限使用期限。

还可通过使用口令失效日期策略（**pwdexpdate** 属性）来设置精确的口令失效日期。

口令失效日期

口令失效日期对特定组织或整个注册表的帐户口令失效时间设置精确日期。

通常，DCE 安全性禁止口令已失效的用户登录。但是，为了防止单元管理员因口令失效而被锁在系统之外，对于如 **cell_admin** 用户可以覆盖这个策略。通过将 *passwd_override* ERA 的实例与主体相连来完成它。有关如何完成这一操作的信息，请参阅第283页的『第30章 创建和维护主体、组和组织』。

以下面的形式，将口令失效日期定义为 **dcecp pwdexpdate** 属性:

pwdexpdate {*date* | **none**}

其中，*date* 是口令失效的日期，它的格式为 *yyyy-mm-dd*，**none** 指定口令无限期。

还可以使用口令使用期限策略（**pwdlife** 属性）设置口令在多久之后失效。

口令格式

口令格式策略适用于特定组织或整个注册表。它们确定下列各项:

- 口令的最小长度，由 **dcecp registry modify pwdminlen** 属性以下列形式定义:

  pwdminlen *integer*

  口令包含的字符数不能少于为 *integer* 输入的数字。如果指定 **0**（零），则最小长度不起作用。
- 口令是否可以完全由空格组成，由 **dcecp pwdspaces** 属性以下列形式定义:

  pwdspaces {yes | no}

  如果指定 **no**，口令不能完全由空格组成。
- 口令是否可以完全由字母数字字符组成，由 **dcecp pwdalpha** 属性以下列形式定义:

  pwdalpha {yes | no}

如果指定 **no**，口令必须至少包含一个非字母数字字符。

注: 通过将 ERA 与主体相连，可以对口令格式施加附加控制。有关如何完成这一操作的信息，请参阅第283页的『第30章 创建和维护主体、组和组织』。

认证策略

认证策略管制票券寿命。可以使用 **dcecp registry modify** 命令为整个注册表设置认证策略，使用 **dcecp account modify** 命令为个别帐户设置认证策略。在以下部分中描述了您可设置的认证策略。

注: 要知道，除了本部分描述的认证策略，还可通过将 *pre_auth_req* ERA 的实例与主体相连来设置主体的控制预认证策略。有关预认证的综合讨论和预认证管理的信息，请参阅第283页的『第30章 创建和维护主体、组和组织』 。

最大的票券可更新时间

注: 当刷新服务票券时，DCE RPC 当前不使用可更新的票券功能。但是， “DCE 安全性服务器” 支持它，它对将 “DCE 安全性服务器” 用作 KDC 的 Kerberos V5 应用程序有用。

您设置的最大票券可更新时间（**maxtktrenew** 属性）确定在主体的票券授予票券失效之前的最大时间量（以小时为单位）和主体必须再次登录以重新认证并获得另一个票券授予票券的时间。设置的最大票券可更新时间越短，系统的安全性就越高。但是，由于用户必须再次登录以更新他们的票券授予票券，所以这个时间需要考虑到用户的方便和单元需要的安全性级别。

以下面的形式，用 **dcecp maxtktrenew** 属性定义最大的票券可更新时间:

maxtktrenew *hours*

其中，*hours* 是一个数字，它表示在主体的票券授予票券失效之前的小时数。

注意，可以使用 **account modify** 命置为个别帐户设置这个时间。

最大的票券寿命

最大的票券寿命（**maxtktlife** 属性）是向主体发出的票券的最大有效小时数。当客户机请求服务器的票券时，授予票券的寿命考虑为服务器和客户机同时设置的最大票券寿命。授予的寿命将不超过服务器和客户机的最大票券寿命中的较小值。

以下面的形式，用 **dcecp maxtktlife** 属性定义最大的票券寿命:

maxtktlife *hours*

其中，*hours* 是一个数字，它表示向主体发出的票券的有效小时数。

设置的最大票券寿命越短，系统的安全性就越高。但是，极其频繁的更新会引起进程的额外开销。您设置的最大票券寿命需要考虑系统性能和您需要的安全性级别。

注意，可以使用 **account modify** 命令为个别帐户设置这个时间。

处理冲突策略

不同的标准和认证策略可以作用于整个注册表和个别组织（对于标准策略）和帐户（对于认证策略）。如果为整个注册表设置的策略不同于为个别组织或帐户设置的策

略，则应用更严格的策略。例如，假定注册表策略指定最小口令长度为 6 个字符，名为 **classic** 的组织的策略指定 8 个字符。如果创建帐户 **bach cantata classic**，则应用更严格的策略（在这种情况下，是组织策略），并且帐户口令长度必须至少为 8 个字符。表25 列出每个策略类型的更严格策略。

表 25. 更严格的标准策略

对于这种策略...	这是更严格的策略...
口令失效日期	更短的失效时间。
口令使用期限	更短的使用期。
帐户使用期限	更短的使用期限。
口令长度	较大的长度。
完全由空格组成的口令	口令不能完全由空格组成；它必须包括一些字符
完全由字母数字组成的口令	口令不能完全由字母数字组成；它必须包括一些非字母数字字符
最大的票券可更新时间	更短的时间（当刷新服务票券时，DCE RPC 通常不使用可更新的票券功能。但是，“DCE 安全性服务器”支持它，它对将“DCE 安全性服务器”用作 KDC 的 Kerberos V5 应用程序有用。）
最大的票券寿命	更短的时间。

创建注册表时，标准策略在缺省情况下处于它们的最许可状态；即，口令失效日期是 **none**，口令和帐户使用期限是 **unlimited**，最小口令长度是 **0**，以及口令可以完全由空格或完全由字母数字组成。最大的票券寿命设置为 10 小时。（通常不使用最大的票券可更新时间。）要实现更严格的策略，必须使用 **registry modify** 命令。

现有策略更新的效果

除了第354页的『口令格式』中描述的口令格式策略，策略更改影响所有现有的帐户和所有在更改之后创建的帐户。

对口令格式策略（如口令长度、口令是否可以完全由空格组成以及口令是否可以完全由字母数字字符组成）的更改，只影响在策略更改之后创建的那些帐户的口令。更改不影响现有的口令。例如，如果更改最小口令长度策略来执行长度更大的口令，则不会影响长度小于新策略指定的长度的现有口令。不需要更改它们，但所有创建的新口令必须遵守新策略。然而，下次更改现有口令时，将执行长度更大的策略。

显示和设置标准和认证策略

要显示策略:

- 对于整个注册表，使用带 **-policies** 选项的 **dcecp registry show** 命令。
- 对于个别组织或帐户，使用带 **-policies** 选项的 **dcecp organization show** 命令（对于标准策略），或使用带 **-policies** 选项的 **dcecp account show** 命令（对于认证策略）。

要设置策略:

- 对于整个注册表，使用 **dcecp registry modify** 命令。下面的样本命令使用 **pwdlife** 选项，将整个注册表的口令使用期限策略设置为 180 天:

  ```
  dcecp> registry modify -pwdlife 180
  dcecp>
  ```

- 对于个别组织或帐户，使用 **dcecp organization modify** 命令（对于标准策略）或 **dcecp account modify** 命令（对于认证策略）。下面的样本命令使用 **-pwdlife** 属性选项，将组织 **classic** 的口令使用期限策略设置为 **unlimited**:

  ```
  dcecp> organization modify classic -pwdlife unlimited
  dcecp>
  ```

注意，前面的例子都使用属性选项。也可以通过使用带 **-change** 选项的 **dcecp registry modify**、**dcecp account modify** 和 **dcecp organization modify** 命令和属性列表来设置策略。例如，要使用属性列表，将组织 **classic** 的口令使用期限策略设置为 **unlimited**，命令如下所示:

```
dcecp> organization modify classic -change {pwdlife unlimited}
dcecp>
```

特性

dcecp registry modify 命令设置整个注册表的特性。在以下部分中描述了您可设置的特性。

缺省票券寿命特性

缺省票券寿命是向注册表中主体发出的票券的缺省寿命（以小时为单位）。

以下面的形式，用 **dcecp deftktlife** 属性设置缺省票券寿命:

deftktlife *hours*

其中，*hours* 是一个数字，表示寿命的小时数。

隐藏口令特性

隐藏口令特性确定是否显示加密口令。以下面的形式，用 **dcecp hidepwd** 属性设置隐藏口令特性:

hidepwd
{yes | no}

其中，**yes** 表示在命令输出和文件中口令出现的地方显示一个 ***** （星号），而不显示加密口令，**no** 表示显示隐藏口令。

最小组标识特性

最小组标识特性是当组的帐户添加到注册表时 “DCE 安全性服务” 自动生成的组标识的起始点。（可以明确地输入一个比这个数小的组标识；它仅适用于自动生成的数。）

以下面的形式，用 **dcecp mingid** 属性设置最小组标识特性:

mingid *integer*

其中，*integer* 是起始标识号。

最小组织标识特性

最小组织标识特性是当组织的帐户添加到注册表时安全性服务自动生成的组织标识的起始点。(可以明确地输入一个比这个数小的组织标识；它仅适用于自动生成的数。)

以下面的形式，用 **dcecp minorgid** 属性设置最小组织标识特性:

minorgid *integer*

其中，*integer* 是起始标识号。

最小 UNIX 标识特性

最小 UNIX 标识特性是当主体的帐户添加到注册表时安全性服务自动生成的 UNIX 标识的起始点。(可以明确地输入一个比这个数小的 UNIX 标识；它仅适用于自动生成的数。)

以下面的形式，用 **dcecp minuid** 属性设置最小组织标识特性:

minuid *integer*

其中，*integer* 是起始标识号。

最大的 UNIX 标识特性

最大的 UNIX 标识特性（**maxuid** 属性）允许您设置在主体的帐户时作为 UNIX 标识提供的最大号。这个最大号适用于系统生成的和用户输入的UNIX 标识。

以下面的形式，用 **dcecp maxuid** 属性设置最大的 UNIX 标识特性:

maxuid *integer*

其中，*integer* 是起始 UNIX 标识。

最小票券寿命特性

最小票券寿命是以分钟为单位的最小时间量，在这段时间之前必须更新主体的票券。这种更新是自动执行的，不会干涉用户。设置的最小票券寿命越短，系统的安全性就越高。但是，极其频繁的更新会降低系统性能。您设置的最小票券寿命需要考虑系统性能和单元需要的安全性级别。

以下面的形式，用 **dcecp mintktlife** 属性设置最小票券寿命:

mintktlife *integer*

其中，*integer* 是一个数字，它表示最小票券寿命的分钟数。

最小票券寿命只能设置为注册表特性。它不能为个别帐户设置。(将它与用 **dcecp registry modify** 或 **account modify** 命令设置的最大票券寿命特性对比。)

显示和设置特性

要显示注册表特性，可使用 **dcecp registry show** 命令。

要设置注册表特性，可使用 **dcecp registry modify** 命令。下面的样本命令使用 **maxuid** 选项，将最大的 UNIX 标识特性更改为 67899:

```
dcecp> registry modify -maxuid 67899
dcecp>
```

注意，上面的例子使用一个属性选项。也可以通过使用带 **-change** 选项的 **dcecp registry modify** 命令和属性列表来设置特性。例如，要使用一个属性列表，将最大的 UNIX 标识特性设置为 67899，命令将是

```
dcecp> registry modify -change {maxuid 67899}
dcecp>
```

第36章 执行例行维护

本章描述应该按照规则基础执行的安全性维护过程，如:

- 将新用户添加到注册表
- 为个别机器创建覆盖
- 更改主密钥
- 备份和恢复数据库
- 更新 **/etc/passwd** 和 **/etc/group** 文件，使它们与注册表一致

添加帐户

要将新用户帐户添加到注册表，必须对注册表有适当的权限（请参阅第419页的『第43章 访问注册表对象』）。一旦有适当的权限，就可以按照下列步骤添加帐户:

1. 如果要在帐户中使用的主体还不存在，则执行 **principal create** 命令来添加主体。
2. 如果将在帐户中使用的组还不存在，则执行 **group create** 命令来添加它。
3. 如果将在帐户中使用的组织还不存在，则执行 **organization create** 命令来添加它。
4. 最后，执行 **account create** 命令来添加帐户。

覆盖本地注册表中的项

可以覆盖本地机器的注册表项。通过使用覆盖，可以防止个人和组登录到特殊机器、建立本地 root 口令以及设计本地用户环境。覆盖信息只对本地机器起作用，而不影响存储在注册表中的帐户信息。

覆盖机制提供一种高级别的本地自主性并允许个别用户控制他们自己的机器。例如，负责一组机器的管理员可以使用覆盖工具来限制对这些机器的访问。管理员可以允许访问特定组，或者管理员可以允许访问除特定组或主体外的每个人。

覆盖的工作原理

存储在本地机器的 **/opt/dcelocal/etc** 目录中的 **passwd_override** 管理文件包含覆盖信息。通过使用这个文件，可以为以下各项输入覆盖:

- 口令
- GECOS 信息
- 主目录
- 登录外壳
- 组成员
- 主体的 UNIX 标识

您输入的覆盖信息只对存储 **passwd_override** 文件的本地机器起作用。当用户使用覆盖文件登录到机器时，覆盖文件中用户帐户的所有信息会替换从注册表获得的相关信息。

例如，假设 **bach** 的注册表帐户在登录时指定 Korn 外壳。由于 **bach** 通常登录到可以运行 Korn 外壳的机器，所以这有利于多数情况。然而，**bach** 有时候为另外部门工作并登录到不能运行 Korn 外壳的机器。要适应 **bach** 的需要，可以在不能运行 Korn

外壳的机器上创建覆盖文件。覆盖可以指定 Bourne 登录外壳。然后，如果 **bach** 登录到可以运行 Korn 外壳的机器，则使用注册表数据并调用 Korn 外壳。当 **bach** 登录到不能运行 Korn 外壳的机器，则使用覆盖数据并调用 Bourne 外壳。

注: 覆盖服务不能用于 DCE 瘦客户机。覆盖服务由未经配置或正在瘦客户机上运行的 **dced** 守护进程提供。

passwd_override 文件格式

passwd_override 文件中的项有下列格式:

```
principal_name:passwd:principal_uid:group_uid:GECOS:home_dir:shell
```

其中:

principal_name

包含主体名的关键字段，该主体名标识覆盖应用于其帐户的主体。输入 *principal_name*，只将覆盖应用于主体的主名称的帐户，而不应用于主体的别名的帐户。

必须输入关键字段 *principal_name*、*principal_uid* 或 *group_uid* 中的任何一个来标识覆盖所应用的帐户。

passwd 加密口令。如果指定覆盖，则在此输入的口令只影响该本地机器。

还可以在 *passwd* 字段中指定 **OMIT**，禁止在本地机器上登录。将 **OMIT** 与 **passwd_export** 命令的选项结合使用，可防止由 **passwd_export** 创建的口令文件中包含用户。（请参阅第366页的『省略本地口令文件中的用户』。）

principal_uid

principal_uid 字段，它包含指定主体的本地标识的 UNIX 标识，当没有输入 *principal_name* 关键字段时，它可起到关键字段作用，或者当结合 *principal_name* 输入时，它可起到包含覆盖的字段作用。

当要将覆盖应用于主体的所有帐户（包括主体的别名的所有帐户）时，可输入 *principal_uid*，而不是 *principal_name*。*principal_uid* 关键字段对至 **root** 的覆盖特别有用。例如，如果 **root** 的别名是 **virtuoso**，则仅当 root 以 **root** 身份登录时，才应用由主体名键入的覆盖。当 root 以 **root**、**virtuoso** 和任何其它别名身份登录时，应用由 root 的 *principal_uid* 键入的覆盖。

输入 *principal_uid* 和 *principal_name* 来覆盖已命名的主体的 UNIX 标识。

group_uid

group_uid 字段，它包含组 UNIX 标识，当没有输入其它关键字段时，它可起到关键字段作用，或者当结合 *principal_name* 或 *principal_uid* 输入时，它可起到包含覆盖的字段作用。

输入 *group_uid* 而不输入其它关键字段（*principal_name* 或 *principal_uid*），将覆盖应用于用 *group_uid* 标识的组的所有成员。在这个实例中，*group_uid* 字段起到关键字段作用，标识要应用覆盖的帐户；即，其主体是指定组成员的帐户。

输入 *group_uid* 和 *principal_name*，将由 *principal_name* 标识的主体的组更改为由 *group_uid* 标识的组。更改只应用于主体主名称的帐户，而不应用于主体别名的任何帐户。输入 *group_uid* 和 *principal_uid*，将组覆盖应用于主体的所

有帐户（包括主体别名的所有帐户）。在这些实例中，*group_uid* 字段起到提供覆盖信息的字段作用，而不是起关键字段作用。

GECOS
帐户的 GECOS 字段。如果指定一个覆盖，它会反映在通过 UNIX **finger** 命令显示的信息中。

home_dir
帐户的主目录。如果指定一个覆盖，则指定的目录只是该机器上帐户的主目录。

shell 当帐户登录时调用的外壳。如果指定一个覆盖，则在登录到该机器时调用您所指定的外壳。

group_override 文件格式

本节说明 **group_override** 文件。

描述

/opt/dcelocal/etc/group_override 管理文件允许您覆盖组的 UNIX 组标识，其方法类似于 **passwd_override** 文件允许覆盖网络注册表数据库中的信息。

group_override 文件存储在每台机器上。对它所做的任何更改都只影响本地机器；它们不影响集中式注册表。如果与注册表一起提供的缺省组定义与本地 UNIX 系统不匹配，则您可能会发现使用文件 **group_override** 对覆盖它们特别有用。

文件格式

文件 **group_override** 中项的格式与 UNIX **group** 文件中项的格式相似。这个格式是:

```
group_name:passwd:group_uid:members
```

在这个项中，*group_name* 和 *group_uid* 是关键字段。您必须输入一个字段来标识要应用覆盖的组。当使用 **passwd_export** 命令时，关键字段用于执行覆盖文件中的查找。查找是按照覆盖文件中指定的项次序来执行的: 先按照组名，然后按照组 UNIX 标识。如果在一个覆盖项中指定两个关键字段，则组名用作查找键；后面的字段用作覆盖。

字段描述

下面的列表描述文件 **group_override** 中的每个项:

group_name
包含名称的关键字段，该名称标识要应用覆盖的组。

passwd 这个字段指定加密口令。如果在这个字段中指定覆盖，则输入的口令只影响该本地机器。

将 **OMIT** 与 **passwd_export** 命令的选项一起使用，可防止由 **passwd_export** 命令创建的组文件中包含这个组。这有效地禁止对本地机器上的这个组执行 **newgrp** 命令。（有关详细信息，请参阅 第364页的『使用 OMIT』。）

group_uid
UNIX 组标识。当没有输入其它关键字段时，这个字段可以起到关键字段作用。当与 *group_name* 一起输入时，它还可起到包含覆盖的字段作用。*group_uid* 值指定由网络注册表服务器提供的组标识的本地覆盖。

members
这个字段指定用逗号分隔的组成员列表。当 **passwd_export** 命令创建

/etc/group 文件时，这个字段的内容将覆盖注册表中的信息。注意，要指定一个空成员（相反地表示无需覆盖），可为这个字段使用一个 *****（星号）。

使字段保留空白

如果不想覆盖一个项，则使其字段保留为空白，并用 :（冒号）分隔每个空白字段。注意，要覆盖带空成员列表的组，可为 *members* 字段输入一个星号。

使用 OMIT

如果指定 **OMIT** 并发出带 **-x** 选项的 **passwd_export** 命令，则已命名的组将不会出现在通过 **passwd_export** 命令产生的 **/etc/group** 文件中。随后，用户将无法对本地机器上的这个组发出 **newgrp** 命令。

同样地，**ls** 命令也受到影响。例如，下面的命令访问组文件以获得有关组的附加信息:

```
ls -lg
```

如果忽略组，则不存在组项并且没有任何可用的信息。由于这个原因，所以仅当用户团体非常大并出现下列其中一种情况时，才应该使用 **OMIT** 来忽略文件 **/etc/group** 中的组:

- 组文件占据的空间太大。
- 组标识到名称的映象太慢（例如，在 **ls -lg** 命令期间）。

例子

要将组 **kmem** 的组标识覆盖为 **3**，可使用下列项:

```
kmem::3:
```

要将组 **system** 的组口令和成员覆盖为单一帐户 **root**，可使用下列项:

```
system:*::root
```

创建覆盖文件项

要创建覆盖文件项，可编辑 **passwd_override** 文件并提供覆盖项。通过指定下列其中一个关键字段，项必须标识要应用覆盖的一个或多个帐户:

principal_name
: 要应用覆盖的特定主体的名称。覆盖仅应用于主体的主名称的帐户。例如，如果将 **mahler** 指定为覆盖项中的主体名，则覆盖仅应用于主体 **mahler** 的帐户，而不应用于 **mahler** 的别名的任何帐户。

principal_uid
: UNIX 标识，它标识当未指定 *principal_name* 时要应用覆盖的帐户。覆盖应用于由 *principal_uid* 标识的主体的所有帐户，包括主体别名的任何帐户。例如，假设主体 **mahler** 的 UNIX 标识为 **2195**。如果将 **2195** 指定为项的键，则覆盖应用于与该 UNIX 标识相关的所有帐户。因为主体的主名称和别名有同一个 UNIX 标识，这意味着，覆盖应用于主体的主名称和所有别名的帐户。

group_uid
: 一个 UNIX 标识，它标识当既没有指定 *principal_name* 也没有指定 *principal_uid* 时要应用覆盖的组。覆盖应用于隶属于标识组的所有主体的所有帐户。

principal_name 字段总是担当关键字段并且不能覆盖。如果输入 *principal_name*，则它标识要覆盖的特定帐户。 *principal_uid* 或 *group_uid* 字段可担当关键字段，或者它们可

以担当覆盖字段。只有其中的一个关键字段才可能用做任何一个项的键；其它字段（如果已输入）都用做覆盖字段。*principal_name* 字段最优先，接着是 *principal_uid*，最后是 *group_uid*。

例如，如果输入 *principal_uid* 而不输入 *principal_name*，则 *principal_uid* 用做关键字段。如果输入 *principal_uid* 和 *principal_name*，则 *principal_name* 用做关键字段，*principal_uid* 用做覆盖字段。

如果输入 *group_uid* 字段而不 输入任何其它关键字段（*principal_name* 或 *principal_uid*），则 *group_uid* 用做关键字段，覆盖应用于组的所有成员的帐户。如果输入 *principal_name* 和 *group_uid*，则 *principal_name* 用做关键字段，已命令主体的组从属关系由用UNIX 标识标识的组覆盖。

使 passwd_override 文件字段保留空白

如果不想覆盖 **passwd_override** 文件中的项，则使其字段保留为空白，并用:（冒号）分隔每个空白字段。然而，必须输入任何一个关键字段来标识正在为其创建覆盖的主体或组。例如，覆盖由 **mozart** 标识的帐户的主目录类似于:

```
mozart:::::/aria/wolfgang:
```

必须输入与任何空白尾随字段相关的冒号。在前面的例子中，外壳字段需要冒号，它是主目录字段后面的剩余字段。

指定特定机器的口令

在本地机器上的 **passwd_override** 文件中手工编辑口令项，以在覆盖文件中创建一个项，用于覆盖本地机器上的口令。输入的口令必须是加密的，但可以从 **/etc/passwd** 文件复制加密口令或者可以编写生成加密口令的程序。

当覆盖主体的口令时，登录时只获得主体的本地凭证，而没有获得主体的网络凭证。如果没有网络凭证，主体就不能访问网络注册表并获得通常在网络登录时提供的信息。因此，必须在 **password_override** 文件项中提供所有这些信息。对于口令的覆盖，必须输入覆盖项中的所有字段，包括所有关键字段。

下面的例子显示 **passwd_override** 文件项，它更改用户 **mozart** 帐户的特定机器口令:

```
mozart:sq1Rc1Urrb1L6:678:893:Wolfgang A. Mozart:/aria/wolfgang:/bin/csh
```

注: 如果口令被覆盖，然后使用 **rlogin** 或 **rsh** 远程登录带覆盖的机器，不管 **/etc/hosts.equiv** 或 **.rhosts** 文件中有些什么，都会提示您输入口令。

防止登录到机器

要防止用户登录到机器，可创建一个覆盖项，其 **passwd** 字段是无效字符串。因为 **passwd** 字段包含加密口令，所以任何长度不是 13 个字符的字符串可以用作无效口令。例如，以下 **passwd_override** 文件中的项将 **exclude** 作为口令提供。长度小于 13 个字符的字符串防止 UNIX 标识为 **25** 的组成员登录。

```
:exclude::25:::
```

省略本地口令文件中的用户

passwd_override 文件中的无效口令项阻止用户登录到文件所在的机器。但是，无效项 **OMIT** 有特殊含意。就象任何其它无效口令那样，如果输入 **OMIT**，用户就不可以登录。另外，如果维护标准 **/etc/passwd** 和 **/etc/group** 文件并使用 **passwd_export** 命令，使这些文件符合注册表数据库，则可以指定口令为 **OMIT** 的用户不包括在 **/etc/passwd** 文件中。（请参阅 第369页的『确保一致的本地文件』，获得有关 **passwd_export** 命令的详细信息。）

同时要知道，如果省略 **/etc/passwd** 文件中的用户，则有关这些用户的信息不可用于使用口令文件的任何程序。例如，**ls -l** 和 **finger** 命令同时访问口令文件来获得由 UNIX 标识标识的用户的进一步信息。如果省略用户，则不存在口令项，而且该用户没有可用的信息。

指定机器的主目录和登录外壳

要更改特定机器的帐户主目录和注册外壳，可创建具有主目录名和注册外壳名的覆盖项。例如，下面的项更改用户 **mozart** 帐户的主目录和注册外壳:

```
mozart:::::rondo/mozart:/bin/ksh
```

覆盖主体的组从属关系

要覆盖主体的组从属关系，可创建一个覆盖项，它包含作为键的主体名或 UNIX 标识和要用作覆盖的组的 UNIX 标识。将主体名用作键，以将覆盖仅应用于主体主名称的帐户。将主体的 UNIX 标识用作键，以将覆盖应用于所有主体帐户，包括主体别名的所有帐户。例如，下面的项覆盖通常与主体 **mozart** 的帐户相关的组:

```
mozart:::356:::
```

这个覆盖不应用于 **mozart** 的任何别名的所有帐户。要将覆盖应用于这些帐户，必须用 **mozart** 的 UNIX 标识 (**567**) 键入该项，如下所示:

```
::567:356:::
```

将覆盖应用于所有组成员

要将覆盖应用于特定组的所有成员，创建一个覆盖项，它包含作为键的组的 UNIX 标识和要覆盖的项。对于组，可以覆盖口令、GECOS 信息、主目录和外壳。例如，下面的项是由 UNIX 标识 **356** 标识的组的所有成员的 **/sonata/piano** 主目录:

```
:::356::/sonata/piano:
```

确保没有包括 *principal_name* 或 *principal_uid* 关键字段。确保之后，提供的主体名或 UNIX 标识将用作关键字段，组 UNIX 标识将用于覆盖该主体的组从属关系。

passwd_override 如何处理多个覆盖项

当多个覆盖项应用于一个帐户时，会选择具有最特定的帐户标识（即，主体 UNIX 标识、组 UNIX 标识或主体名）的项。主体名最特定，接着是主体 UNIX 标识和组 UNIX 标识。

例如，假设覆盖文件包含下列两个覆盖注册外壳的项:

```
mozart::::::/bin/ksh
:::25:::/bin/csh
```

如果主体以 **mozart** 身份登录，则由 **mozart** 键入的覆盖是有效的。在这种情况下，主体（**mozart**）比组（**25**）更特定。

更改注册表的主键

注: 有关 “DCE 安全性注册表” 和 “LDAP 集成” 功能的该主题信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

存储在注册表中的所有口令都由主键进行加密。注意，主键是在系统配置期间创建注册表数据库时创建的。

可以使用带 **-key** 选项的 **dcecp registry modify** 命令来更改注册表的主键并用新主键重新加密所有口令。每个（主和从）副本都维护它自己的主键来访问其注册表副本中的数据。

您应该根据常规标准来更改每个副本的主键。在运行任意一个程序来完成这一任务之前，请确保登录到管理帐户。

下列的命令行更改主键并重新加密副本 **art_server_1** 的所有口令:

```
dcecp> registry modify /.../giverny.com/subsys/dce/sec/art_server_1
-key
dcecp>
```

验证 DCE 安全性服务的可靠性

DCE 守护程序中的 **secval** 进程可以确认 DCE 安全性服务器是一个可信的服务器。违法的 DCE 安全性服务器可能通过返回一个伪造的本地系统身份来对机器进行恶意的用户 root 访问。通过对 **secval** 进程执行一个经认证的 RPC，**secval ping** 操作确认 DCE 安全性服务器的可靠性。成功返回（**1**）表示安全性服务器使用了认证的 RPC 获得成功所需的所有正确口令。

可以对本地主机执行 **secval ping** 操作，或对远程主机提供一个要操作的变量。因为远程主机可能使用不同的安全性服务器，所以对远程主机执行 **secval ping** 操作时，会提供测试单元中其它安全性服务器的可靠性的方法。

下面的例子说明对远程主机 **charon** 上的 **secval** 进程执行 **secval ping** 操作:

```
dcecp> secval ping /.:/hosts/charon/config/secval
1
dcecp>
```

备份和恢复注册表数据库

注: 有关 “DCE 安全性注册表” 和 “LDAP 集成” 功能的该主题信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

使用在此描述的准确过程来备份注册表数据库，防止备份在备份期间到达主副本。

只需要备份主副本数据库及其主键文件。当备份存储主副本及其主键的整个磁盘时和当只备份主副本的数据库文件及其主键文件时，使用在以下部分在描述的过程。

备份注册表数据库的过程

注: 有关 “DCE 安全性注册表” 和 “LDAP 集成” 功能的该主题信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

要运行备份过程，请确保经由管理帐户登录到 DCE。然后，运行 DCE 控制程序来进行备份。备份步骤如下:

1. 输入 **registry disable** 命令，将主副本设置为维护状态。以下命令将单元 **giverny.com** 中的主注册表设置为维护状态:

   ```
   dcecp> registry disable  /.../giverny.com/subsys/dce/sec/oddball
   dcecp>
   ```

   如果将主副本设置为维护状态，可使主副本将它的数据库保存到磁盘并拒绝所有更新。

2. 通过备份整个卷或 *dcelocal*/**var/security/rgy_data** 树（注册表）和 *dcelocal*/**var/security/.mkey** 文件（它包含用于加密注册表中所有键的主键）来备份主注册表。注意，因为 *dcelocal*/**var/security/.mkey** 文件包含主键，所以恢复注册表数据库的备份是无用的，除非还恢复了 *dcelocal*/**var/security/.mkey** 文件。

   用于备份的精确命令对于个人首选项很重要。但是，如果将数据库和主密钥文件写入同一个磁带，则将磁带储藏在带受限访问权的锁定区域中。或者，可以将数据库和密钥文件写入单独磁带，并将每个磁带储藏在不同地方。

3. 当备份完成时，使主副本脱离维护状态，如下所示:

   ```
   dcecp> registry enable  /.../giverny.com/subsys/dce/sec/oddball
   dcecp>
   ```

   安全性服务器继续接受更新。

注意，前面的例子为 **registry enable** 和 **registry disable** 命令提供了注册表主位置的名称。如果没有提供注册表位置名，命令使用在 **_s(sec)** 变量中命名的位置。如果没有设置这个变量，命令使用机器的缺省单元的主注册表。有关详细信息，请参阅第369页的『设置 _s(sec) 变量』。

恢复注册表数据库的过程

注: 有关 “DCE 安全性注册表” 和 “LDAP 集成” 功能的该主题信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

本部分提供恢复主副本的数据库文件和主键文件的介绍。过程假设正在将数据库恢复到备份它的相同机器并且正在使用 DCE 控制程序。如果正在将数据库移动到不同机器，请遵循第371页的『第37章 处理网络重新配置』中的指令。

要将注册表数据库恢复到机器，请执行下列步骤:

1. 在主注册表位置以 **root** 身份登录。
2. 如果 **secd** 正在运行，发出 **registry stop** 命令来停止它。当使用这个命令时，必须将特定副本的全限定名作为变量提供。下列样本命令停止名为 **oddball** 的 **secd**:

   ```
   dcecp> registry stop /.../giverny.com/subsys/dce/sec/oddball
   dcecp>
   ```

3. 将备份文件从备份媒体复制到机器。如果只备份了注册表数据文件和主键文件，请确保将注册表数据库复制到 *dcelocal*/**var/security/rgy_data**，将主键文件复制到

dcelocal/var/security/.mkey。注意，因为 *dcelocal*/var/security/.mkey 文件包含主键，恢复注册表数据库的备份是无用的，除非还恢复了 *dcelocal*/var/security/.mkey 文件。

4. 通过调用带 **-restore_master** 选项的 **secd** 来重新启动服务器，如下所示:

   ```
   dcelocal/bin/secd -restore_master&
   ```

   这个命令启动 **secd** 并使主副本将所有从属副本标记为要重新初始化。

5. 验证在系统启动时 **secd** 是否自动启动。

注: 如果只恢复主键文件并且没有更改主键，则可以只从备份媒体复制主键文件，而无需执行恢复过程中的所有其它步骤。

设置 _s(sec) 变量

可以将要绑定到的注册表位置的名称作为对注册表进行操作的 **dcecp** 命令的变量提供。如果没有提供名称，命令绑定到在 **_s(sec)** 变量中命名的副本。如果没有设置这个变量，则命令绑定到单元的主副本。可以设置 **_s(sec)** 变量，然后将该副本用作 **dcecp registry** 命令的缺省副本。要完成这个操作，可使用下面样本中显示的 **set** 命令，它将缺省副本设置为单元 **giverny.com** 中的主副本（名为 **slave_3**）:

```
dcecp> set _s(sec) /.../giverny.com/subsys/dce/sec/slave_3
dcecp>
```

作为 **set** 命令的变量提供的新缺省副本的名称可以是下列任一形式:

- 单元名（例如，**/.../dresden.com**）

  如果输入单元名，已命名的单元成为缺省单元。DCE 控制程序随意选择已命名单元中要绑定到的副本，该副本成为缺省副本。

- 创建副本时给定的全局名（例如，**/.../dresden.com/subsys/dce/sec/rs_server_250_2**）

  全局名标识特定单元中的特定副本。该单元成为缺省单元，而该副本成为缺省副本。

- 当前缺省副本的副本列表中出现的副本名（即，其与单元相关的名称；例如，**subsys/dce/sec/rs_server_250_2**）

  该副本成为缺省副本，而该副本所在的单元成为缺省单元。

- 正在运行副本的主机的网络地址（例如，**ncadg_ip_udp:15.22.144.248**）

  该主机上的副本成为缺省副本，而该主机所在的单元成为缺省单元。

某些 **dcecp** 命令可以只作用于主副本，因而，需要绑定到主副本。如果执行只作用于主副本的命令而且该主副本不是缺省副本，在多数情况下， **dcecp** 自动尝试绑定到当前缺省单元中的主副本。在其它情况下， **dcecp** 显示一条错误消息，命令失败。

确保一致的本地文件

passwd_export 命令使本地机器上的标准 **/etc/passwd** 和 **/etc/group** 文件与注册表数据库一致。最好根据有规律而交错的原则运行 **passwd_export** 命令，作为 **cron** 操作的一部分。如果 **passwd_export** 在创建新口令和组文件时成功，它将当前文件另存为名为 **passwd.bak** 和 **group.bak** 的备份。如果它失败，它会使当前文件保持原样。

passwd_export 命令的语法如下:

```
passwd_export
[-n ][-d  directory_name] [-x ] [-m max_entries] [-s ] | [-h [elp] [-v ]
```

其中:

-n 指定应该忽略 **passwd_override** 和 **group_override** 文件项。如果没有这个标志，**passwd_export** 将两个文件中的覆盖项应用于它所创建的本地口令和组文件。

-d *directory_name*
指定要存储由 **passwd_export** 创建的本地口令和组文件的目录的名称。如果没有输入这个选项，缺省情况下，这些文件存储在本地节点上的 **/etc** 目录中。

例如，要将文件存储在名为 **/etc/locals** 的目录中，可以下列形式输入该命令:

```
dceshared/bin/passwd_export -d/etc/locals
```

-x 对于具有将 **OMIT** 指定为其加密口令的口令或组覆盖（在本地机器上）的用户，禁止项的创建。使用 **-x** 选项，使被省略的用户或组不包括在由 **passwd_export** 创建的口令和组文件中。要省略用户，必须为用户创建一个覆盖项，并输入单词 **OMIT** 作为用户的口令字段项。被省略的用户无法登录到本地机器。（请参阅第366页的『省略本地口令文件中的用户』。）要省略组，可为组创建一个覆盖项，并输入单词 **OMIT** 作为组的口令字段项。

-m *max_entries*
设置放入 **/etc/passwd** 和 **/etc/group** 文件的最大注册表项数。

-s 根据 UNIX 号来排序 **/etc/passwd** 和 **/etc/group** 文件中的项。如果没有指定这个选项，项将按照在注册表中的随机检索次序排序。

-h[elp]
显示帮助信息。

-v 以详细模式运行。

第37章 处理网络重新配置

本章描述处理网络重新配置的过程，以更改注册表副本的位置。明确地讲，本章包括下列内容:

- 更改主注册表位置
- 除去网络中的节点
- 处理网络地址更改

要执行本章中的过程，必须经由管理帐户登录到网络注册表帐户。

更改主副本位置

运行主副本服务器的机器必须总是可用的。如果打算从网络除去这台机器或长期关闭它，则需要更改主副本的位置。

更改主注册表位置的首选方法是，使用 **dcecp registry designate** 命令来颠倒主服务器和从属服务器的角色。换言之，使主服务器成为从属服务器，使从属服务器成为主服务器。有关详细信息，请参阅第399页的『当前主副本出现故障时指定一个新的主副本』。

当调用 **dcecp registry designate** 命令时， 发生下列情况:

1. 当前主服务器将所有暂挂的更新及其传播队列发送到指定为新主服务器的副本。
2. 指定的新主服务器读取当前主服务器的副本列表，以获得它管理至从属服务器的传播队列所需的信息。
3. 当指定的新主服务器获得了当前主服务器的所有必需信息时，它成为新的主服务器，而当前主服务器成为从属服务器。

因为这种有序且完整的信息传送确保没有数据丢失，所以 **dcecp registry designate** 命令是当主位置和从属位置上的注册表服务器正常操作时将主注册表移到另一台机器的首选方法。注意，**dcecp registry designate -master** 命令还可用于将副本从从属副本更改为主副本。然而，因为 **dcecp registry designate -master** 命令会使数据丢失，所以仅在当前主副本已毁坏时才使用它。当由于网络故障而不能到达主服务器或因为主服务器已临时关闭时，建议不要使用它。有关在当前主服务器发生故障或已毁坏时如何指定一个新的主安全性副本的详细信息，请参阅第399页的『当前主副本出现故障时指定一个新的主副本』。

按照这些步骤来更改主副本的位置:

1. 选择新的主副本位置。在这个位置上必须存在一个从属副本。如果有必要，使用 **config.dce** 命令配置一个从属机器。
2. 发出 Tcl **set** 命令，将缺省副本设置为当前主副本。在下面的例子中，主副本设置为单元 **giverny.com** 中名为 **oddball** 的副本:

   ```
   dcecp> set _s(sec) /.../giverny.com/subsys/dce/sec/oddball
   dcecp>
   ```

3. 发出 **registry designate** 命令，颠倒主副本和从属副本的角色。这个命令将要成为新主副本的副本名称作为一个变量。下面的例子使名为 **/.../giverny.com/subsys/dce/sec/music** 的副本成为新主副本:

```
dcecp> registry designate /.../giverny.com/subsys/dce/sec/music
dcecp>
```

4. 验证主位置是否更改过。通过发出 **registry show -replica** 命令来完成它。

除去网络中的服务器设备

注:

1. 有关 “DCE 安全性注册表” 和 “LDAP 集成” 功能的该主题信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。
2. 以下是用于删除副本的手册指令。仅当要删除的副本已成为永久不可用或毁坏时，才遵循这些指令。对于 DCE 3.2，建议在具有要删除的副本的主机上使用 **unconfig.dce** 命令来删除副本。用 **unconfig.dce** 指定 **sec_rep** 组件，以取消配置副本。

如果正打算从网络除去运行从属副本的机器或要长期关闭机器，则删除该位置上的副本。

如果正在除去运行主服务器的节点，则在除去节点之前必须如前面所描述的那样更改主服务器位置。

使用 **dcecp registry delete** 命令来删除从属副本。当执行这个命令时，主副本执行下列操作:

1. 将副本标记为已删除。
2. 将删除传播到其副本列表上的所有副本。
3. 将删除请求发送到副本。
4. 从其副本列表除去副本。

下面的样本命令删除名为 **/.../giverny.com/subsys/dce/sec/art_1** 的从属副本:

```
dcecp> registry delete /.../giverny.com/subsys/dce/sec/art_1
dcecp>
```

当发出这个命令时，**dcecp** 绑定到处于当前单元中的主副本（如果必需的话）；然后，主副本命令从属副本删除其自身。

要验证是否已删除从属副本，可发出 **dcecp registry catalog** 命令。当主副本接收到删除从属副本的请求时，从属副本出现在副本列表中，且标记为删除。当实际删除了副本时，它不再出现在列表中。

处理网络地址更改

当 **secd** 启动时，主副本和从属副本可以检测地址更改并且可以对主副本的副本列表和单元名称空间执行必要的更新。通常，您需要处理网络地址更改的操作只是更新 **pe_site** 文件。然而，如果主副本和从属副本的网络地址同时发生更改，则需要您的介入。本部分描述如何更新 **pe_site** 文件和如何处理同时发生的地址更改。

更新 pe_site 文件

只要主副本或从属副本的网络地址发生更改，就必须更新单元中所有机器上的 **/opt/dcelocal/etc/security/pe_site** 文件。在重新启动 DCE 之前必须完成这一操作，尤其是在运行 “安全性服务器 (**secd**)” 的机器上。这个文件存在于单元中的每台机器上， “DCE 安全性服务” 需要它绑定其自身。这个文件包含由单元中每个 “安全性服

务器”使用的每个界面的入口。在“安全性服务器”机器上，该机器的界面信息位于文件的开头。在非“安全性服务器”机器上，入口按照由该机器使用的 ″lan-profile″ 中指定的顺序（首选副本顺序）或随机顺序。

使用下列命令（作为 **root** 用户）来更新 **pe-site** 文件:

```
dcecp -c secval update
```

如果这个命令失败，则手工更新 **/opt/dcelocal/etc/security/pe_site**。

下面是单元 ″my_cell″ 中的样本 **pe_site** 文件，它有两台“安全性服务器”机器:

```
/.../my_cell 5ddfd8c4-eb13-11d4-83a9-09356135aa77@ncacn_ip_tcp:9.53.96.52[]
/.../my_cell 5ddfd8c4-eb13-11d4-83a9-09356135aa77@ncadg_ip_udp:9.53.96.52[]
/.../my_cell 5ddfd8c4-eb13-11d4-83a9-09356135aa77@ncacn_ip_tcp:9.53.97.53[]
/.../my_cell 5ddfd8c4-eb13-11d4-83a9-09356135aa77@ncadg_ip_udp:9.53.97.53[]
```

处理同时发生的地址更改

如果主副本和从属副本的地址同时发生更改，则当主副本和从属副本尝试通知对方已更改的地址时它们彼此无法到达。要避免这一问题，请确保在另一个地址更改前，一个主或从属副本的地址更改被传播到所有副本。更改一个地址。然后，使用 **dcecp registry show -replica** 命令查看主位置和从属位置上的副本列表。当在两个副本列表上显示新地址时，继续下一个网络地址更改是安全的。

如果无法防止主副本和从属副本的网络地址同时发生更改，则要恢复主副本和从属副本间通信的唯一方法是删除从属副本，然后重新创建它。根据您的情况，使用下列其中一种方法来删除从属副本:

- 如果您预料会同时发生地址更改，则当主副本和从属副本仍在通信时，使用 **set** 命令来绑定到主副本，然后使用 **dcecp registry delete** 命令删除从属副本。
- 如果 **secd** 正在主和从属位置上运行，但主副本和从属副本没有进行通信，则先使用 **set** 命令来绑定到从属副本，然后使用 **registry destroy** 命令来毁坏从属副本。然后使用 **set** 来绑定到主副本，使用 **registry delete -force** 命令来除去从属副本的副本列表项。
- 如果 **secd** 未在从属位置上运行或者如果无法绑定到从属位置，则使用第391页的『第40章 故障排除过程』中描述的重新创建副本过程。

第38章 设置注册表

关于“DCE 安全性注册表”和“LDAP 集成特性”的主题信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

本章描述在“遗留 DCE 安全性服务”中设置注册表应采取的步骤。DCE 安装和配置期间 **config.dce** 脚本会自动处理这些步骤中的一部分；剩余的由用户使用 DCE 实用程序和控制程序来执行。设置注册表的步骤如下:

1. 规划安全性服务组件在您网络中的位置。
2. 创建主注册表数据库（系统配置期间通过 **config.dce** 脚本执行）。
3. 启动主副本（系统配置期间通过 **config.dce** 脚本执行）。
4. 填充注册表数据库（通过使用 **dcecp rgy_edit** 命令执行）。
   - 设置策略与特性。
   - 添加名称和帐户。
5. 创建从属数据库并启动从属副本（系统配置期间通过 **config.dce** 脚本执行）。
6. 设置 **cron** 来运行所有基于 DCE 机器上的 **passwd_export** 以确保本地口令和组文件与注册表一致（由用户通过使用标准 UNIX 命令执行）。**passwd_export** 命令在第361页的『第36章 执行例行维护』中描述。

因为注册表使用“单元目录服务”(CDS) 来获取网络资源信息，本章假设将您的网络已配置成适合于 CDS 操作。

规划 DCE 安全性服务组件的站点

配置网络中安全性服务的首要步骤是为注册表的主副本和任何从属副本选择站点。这些站点将运行 **secd** 命令，即安全性服务器。正在运行 **secd** 命令的机器必须总是启动和可用的。特别重要的是: 主副本运行的机器必须在整个网络可用。

运行 **secd** 所需的机器大小取决于平台和操作系统。一般来说，要选择足够大、以容纳注册表数据库以后增长的机器。机器必须有足够的磁盘空间用于注册表数据库和足够的后备存储器，以便进程不会失效。

当运行脚本时，它将配置主副本位置来运行 DCE 主机守护程序(**dced**)（该守护程序为本地主机提供端点映射表器服务）和任何必需的 CDS 服务器。

创建主注册表数据库

当最初配置您单元的安全性服务器时，**config.dce** 脚本调用 *dcelocal***/bin/sec_create_db** 命令来创建主副本。当 **sec_create_db** 创建新的主副本时，它用名称和帐户初始化它的数据库。请注意，您必须是 **root** 用户才能运行 **sec_create_db**。

sec_create_db 命令还创建名为 *dcelocal***/etc/security/pe_site** 的注册表配置文件，该文件包含主副本的单元名和网络地址。如果 CDS 不可用，则该文件为运行在那台机器上的客户机提供 **secd** 服务器的绑定地址。

如果您需要创建一个新的主副本注册表数据库，则可以直接调用 **sec_create_db**。请注意，您必须是 root 用户才能运行 **sec_create_db**。还要注意，重新创建主数据库的情

况非常罕见，但是如果从属数据库发生毁坏，您可能需要重新创建从属数据库。以下各小节描述如何使用 **sec_create_db** 命令。

sec_create_db 命令格式

sec_create_db 命令有以下格式:

```
sec_create_db {-master | -slave} -my[name] my_server_name \
[-k[eyseed] keyseed] [-cr[eator]creator_name] \
[-cu[nix_id] creator_unix_id] [-u[uid] cell_uuid, ] \
[-p[erson_low_unix_id] unix_id]
[-g[roup_low_unix-id] unix_id] \
[-o[rg_low_unix-id] unix_id] [-ma[x_unix_id] unix_id] \
[-pa[ssword] default_password] [-ldap] [-v[erbose]
```

其中:

-master
指定应该创建主副本的数据库。所有其它 **sec_create_db** 选项可以与 **-master** 选项一起使用。

-slave 指定应该创建从属副本的数据库。只有 **-myname**、**-keyseed** 和 **-verbose** 选项才能与 **-slave** 选项一起使用。

-my[name] *my_server_name*
这是您分配给这台机器上的安全性服务器 (**secd**) 的名称。它由名称服务使用来确定这个单元的安全性服务器的位置。

-k[eyseed] *keyseed*
这是一个字符串，您输入该字符串来产生随机密钥发生器，以为您正在创建的数据库创建主密钥。它应该是一个不能被轻易猜出的字符串。主密钥用于加密所有帐户口令。每个副本实例（主或从属）都有其自己的主密钥。可以使用 **dcecp registry modify** 命令或 **sec_admin master_key** 命令来更改主密钥。（有关使用 **dcecp registry modify** 命令来修改主密钥的信息，请参阅第361页的『第36章 执行例行维护』）。如果不输入这个选项，则 **sec_create_db** 将提示您输入。

-cr[eator] *creator_name*
这是注册表创建者的名称。注册表创建者是注册表数据库的初始特权用户。请注意，可以在任何时候使用 **dcecp acl modify** 命令或 **acl_edit modify** 命令给予其它用户以同等特权，以更改注册表的数据库 ACL。当创建注册表时，同时创建了注册表对象的缺省 ACL 项。这些项将最具特权的许可权给予在 **-cr** 选项中命名的主体。如果命名为注册表创建者的主体不是保留名称之一，则 **sec_create_db** 添加主体和那个主体的 帐户。如果不输入这个选项，则注册表数据库的初始特权用户是 **root**。

-cu[nix_id] *creator_unix_id*
这是您指定分配给注册表创建者的 UNIX 号。如果不输入这个选项，则动态分配注册表创建者的 UNIX 号。

-u[uid] *cell_uuid*
这是单元的 UUID。如果不输入这个 UUID，则动态分配它。

-p[erson_low_unix_id] *unix_id*
这是使用 **dcecp registry modify** 命令或 **rgy_edit properties** 命令添加主体时自动生成的 UNIX 标识的起始点。注意可以显式地指派一个小于该号码的 UNIX 标识；这个更小的限制仅适用于自动创建的 UNIX 标识。

-g[roup_low_unix_id] *unix_id*

这是使用 **dcecp registry modify** 命令或 **rgy_edit properties** 命令添加组时自动生成的 UNIX 标识的起始点。注意可以显式地输入一个小于该号码的 UNIX 标识；这个更小的限制仅适用于自动创建的 UNIX 标识。

-o[rg_low_unix_id] *unix_id*

这是使用 **dcecp registry modify** 命令或 **rgy_edit properties** 命令添加组织时由安全性服务自动生成的 UNIX 标识的起始点。注意可以显式地输入一个小于该号码的 UNIX 标识；这个更小的限制仅适用于自动创建的 UNIX 标识。

-ma[x_unix_id] *unix_id*

这是添加主体、组或组织时可以指派为 UNIX 标识的最大号码。不会自动指派比该号码更大的 UNIX 标识，也不能具体地输入比该号码更大的号码。最大的 UNIX 标识将保持不变，直到使用 **dcecp registry modify** 命令或 **rgy_edit properties** 命令更改它为止。

-pa[ssword] *default_password*

这是指派到通过 **sec_create_db** 命令创建的帐户的缺省口令。如果不指定缺省口令，将使用 **-dce-**。注意帐户 **hosts/** *local_host_name/principal_name* **none none**，**krbtgt** */cell_name* **none**，没有将 **nobody none none** 指派成缺省口令而是使用随机生成的口令。

-ldap 表示将在 LDAP 数据库中存储安全性注册表。

-v[erbose]

在详细方式下运行并生成所有活动的详细副本。

sec_create_db 命令运行示例

下列示例显示了运行 **sec_create_db** 命令来创建主数据库和 **sec_create_db** 命令运行时显示的信息。注意因为没有输入 **-k** 选项， **sec_create_db** 命令将提示您输入主密钥字符串。并不显示真实输入的字符串。

```
/work/krb/sec_create_db -v -myname /.../dresden.com/subsys/dce/sec/master
-master
Enter keyseed for initial database master key: <enter up to
1024 characters>
SECD Checkpoint on Tue Sep 27 11:44:12 1994
....  saving rgy
....  saving acct
....  saving person
....  saving group
....  saving org
....  saving replicas
....  saving acl
End SECD Checkpoint on Tue Sep 27 11:44:13 1994
SECD Checkpoint on Tue Sep 27 11:44:15 1994
....  saving rgy
....  saving acct
....  saving person
....  saving group
....  saving org
....  saving acl
End SECD Checkpoint on Tue Sep 27 11:44:17 1994
```

sec_create_db 命令结果

通过 **sec_create_db** 创建的主注册表数据库包含主体、组和第378页的表26 中列出的组织。

表 26. 初始人、组和组织

主体	组	组织
bin	**bin**	none
daemon	**daemon**	--
dce-ptgt	**kmem**	--
dce-rgy	**mail**	--
krbtgt/ *local_cell_name*	**nogroup**	--
hosts/ *local_host***/self**	**none**	--
mail	**system**	--
nobody	**tcb**	--
root	**tty**	--
sys	**uucp**	--
tcb	--	--
uucp	--	--
who	--	--

sec_create_db 命令创建的帐户是:

- **bin bin none**
- **daemon daemon none**
- **dce-ptgt none none**
- **dce-rgy none none**
- **hosts/***local_host***/self none none**
- **krbtgt/***cell_name* **none none**
- **nobody nogroup none**
- **root system none**
- **uucp uucp none**

保存某些最初通过 **sec_create_db** 命令创建的对象并且它们是无法删除的。这些在下表中说明。

- 保存的主体是:
  - **dce-ptgt**
  - **krbtgt/** *cell_name*
  - **dce-rgy**
- 保存的帐户是:
  - **dce-ptgt none none**
  - **krbtgt/** *cell_name* **none none**
  - **dce-rgy none none**

当运行 **sec_create_db** 命令来创建主注册表数据库时，可以命名具有访问注册表最大权限的主体。该人员称为注册表创建者。 如果注册表创建者不是缺省的主体，**sec_create_db** 命令添加帐户 *rgy_creator* **none none**，其中 *rgy_creator* 是命名为注册表创建者的主体。如果不命名注册表创建者， **sec_create_db** 为 **root system none** 帐户指派最大权限的注册表访问。

有一种例外情况，将为通过 **sec_create_db** 命令创建的所有帐户分配随机生成的口令并将这些帐户成无效。这些主体能登录到这些帐户之前必须更改帐户口令并将帐户标

记成有效。可使用 **dcecp account modify** 命令完成该操作。第311页的『第31章 创建和维护帐户』 提供了使用 **dcecp account modify** 命令更改注册表中主体帐户的所有属性包括主体的口令。**dcecp** 还具有随机生成新口令的选项。

但是，例外的情况是为注册表创建者创建的帐户是有效的并将它指定为 DCE 缺省口令 (**-dce-**)。更改缺省口令确保注册表创建者帐户的安全性。

除了通过 **sec_create_db** 命令创建的帐户所暗示的组成员资格外，主体还将成为表27中列出的组的成员。

表 27. 通过 by sec_create_db 命令创建的组成员资格

主体...	是组... 的成员
who	**bin**
root	**system**
	kmen
	tty
sys	**kmem**
mail	**mail**
tcb	**tcb**

第283页的『第30章 创建和维护主体、组和组织』 提供了如何将主体添加到组的指导。

启动主副本

config.dce 创建主副本后启动主副本。要显式地启动主副本 (**secd**)，请完成下列操作:

1. 在将运行主副本的机器上作为 **root** 登录。
2. 运行 **start.dce sec_svr**。

填充新的注册表数据库

一旦创建并启动了主副本，必须通过设置策略和过程并添加帐户来填充数据库。

设置策略与特性

使用 **dcecp registry show** 和 **dcecp registry modify** 命令查看策略和特性并按期望更改它们。

添加帐户

创建了新的注册表后，它将只包含主体、组和组织以及通过 **sec_create_db** 命令添加的初始信息。使用 **dcecp account create** 命令添加站点所需的任何其它名称和帐户。可现在或在以后的任何时间完成该操作。参阅第311页的『第31章 创建和维护帐户』。以获取关于使用 **dcecp** 命令添加帐户的信息。

创建从属副本

创建并启动主副本数据库，填充其数据库后，在从属站点运行 **config.dce** 创建从属副本并启动它们。要创建并启动从属副本， **config.dce** 命令首先确保该站点正在运行 **dced** 和适当的 CDS 服务器。然后它执行下列 **sec_create_db** 命令:

```
dcelocal/bin/sec_create_db -slave
-myname my_server_name
```

首先，该命令为新的从属副本创建一个数据库。该数据库仅包含 stub 文件。然后该命令定位主副本并将新的从属副本添加到主副本列表中。主副本标记要初始化的新副本。最后，**config.dce** 脚本启动 **secd** 命令并确保每次重新启动机器时自动启动它。

必须在每台想要运行从属副本的机器上运行 **config.dce** 命令来配置从属副本。

验证副本是否正在运行

安装并启动了主副本和从属副本后，请执行下列步骤以确保它们正在运行:

1. 获取单元中正在运行的安全性服务器名称列表。

```
dcecp> registry catalog
/.../dc.cell.ch.hp.com/subsys/dce/sec/dce6
/.../dc.cell.ch.hp.com/subsys/dce/sec/dce5
dcecp>
```

2. 查看副本保存的传播信息:

```
dcecp> registry show /.../dc.cell.ch.hp.com/subsys/dce/sec/dce6
-replica
{name /.../dc.cell.ch.hp.com/subsys/dce/sec/dce6}
{type slave}
{cell /.../dc.cell.ch.hp.com}
{uuid 07f5c1dc-80ef-11cf-b60c-0800095f6636}
{status enabled}
{lastupdtime 1996-09-17-13:06:53.000-04:00I-----}
{lastupdseq 0.33826}
{addresses
 {ncadg_ip_udp 15.22.51.49}
 {ncadn_ip_tcp 15.22.51.49}}
{masteraddrs
 {ncadg_ip_udp 15.22.48.183}
 {ncadn_ip_tcp 15.22.48.183}}
{masterseqnum 0.15724}
{masteruuid 90067612-d3f6-11ce-b773-0800095a49d8}
{supportedversion
 secd.dce.1.0.2
 secd.dce.1.1}
dcecp>
```

3. 查看主副本为每个从属副本保存的传播信息:

```
dcecp> registry show /.../dc.cell.ch.hp.com/subsys/dce/sec/dce5
-master
{name /.../dc.cell.ch.hp.com/subsys/dce/sec/dce6}
{uuid 07f5c1dc-80ef-11cf-b60c-0800095f6636}
{type slave}
{addresses
 {ncadg_ip_udp 15.22.51.49}
 {ncadn_ip_tcp 15.22.51.49}}
{propstatus update}
{lastupdtime 1996-09-17-13:06:53.000-04:00I-----}
{lastupdseq 0.33826}
{numupdtogo 0}
{commstate ok}
{lastcommstatus {Successful completion}}
{name /.../dc.cell.ch.hp.com/subsys/dce/sec/dce5}
{uuid 90067612-d3f6-11ce-b773-0800095a49d8}
{type master}
```

```
{addresses
 {ncadg_ip_udp 15.22.48.183}
 {ncadn_ip_tcp 15.22.48.183}}
dcecp>
```

第39章 将 UNIX 帐户导入 DCE

passwd_import 命令创建注册表中基于 **/etc/passwd** 和 **/etc/group** 文件中信息的项。它提供一种方法，可确保使用 “DCE 安全性服务” 的机器和不使用 “DCE 安全性服务” 的机器之间的帐户一致性，并且提供将一个现有 UNIX 用户库添加到注册表的方法。

passwd_import 的工作原理

当 **passwd_import** 处理项时，它将组和口令文件项与注册表项进行比较。它可以找到两种冲突:

名称冲突

当在注册表和组或口令文件中定义相同的名称字符串时，会发生这些冲突。名称 **joe 102** 和 **joe 555** 例示了这种冲突。重复名称可以表示同一个用户或两个不同用户。

UNIX 标识 冲突

当在注册表和组或口令文件中为不同名称的用户定义同一个 UNIX 标识时，会产生这些冲突。名称 **joe 102** 和 **ann 102** 例示了这种冲突。

这些冲突可以单独出现（如前面的例子）或一起出现。例如，注册表项 **joe 102** 和 UNIX 项 **joe 102** 有冲突。当发现冲突时，必须提供用于更改口令和组文件项的信息或通知 **passwd_import** 不要导入该项。 **passwd_import** 命令不更改现有的注册表主体、组或帐户；但是，如果您指定更改的话，它将在注册表中创建基于组和口令文件的新主体、新组和新帐户。

passwd_import 处理步骤

当 **passwd_import** 处理项时，它依次执行下列步骤:

1. 它打开组和口令文件并建立与注册表的连接。
2. 它将组文件项与注册表中的组进行比较。如果没有冲突，它就在注册表中创建与组文件中的组相符合的组。
3. 它将口令文件中的项与注册表中的主体进行比较。如果又没有冲突，它就:
   - 在注册表中创建与口令文件中的项相符合的主体。
   - 将最近创建的主体添加到适当的组中。
   - 为最近创建的主体创建帐户。
4. 它复查组文件并将主体作为它在那里找到的任何附加组的成员添加。

当处理各个步骤时，个别更改注册表。如果指定 **-o** 选项，则 **passwd_import** 将所有最近创建的注册表主体添加到指定组织中。如果没有指定组织，主体将添加到组织 **none** 中。

由 passwd_import 创建的注册表项

如果口令或组文件中存在项而注册表中不存在项，则 **passwd_import** 创建一个新的注册表项。对于附加注册表信息， **passwd_import** 使用下列值:

- 对于 “主体” 和 “组” 项:

- **Alias/Primary Name** = 如果口令文件包含两个具有同一 UNIX 号的项，则 **passwd_import** 为它找到的第一个 UNIX 号创建一个主体名项，并为同一 UNIX 号的每次出现创建一个别名。
- **Full Name** = 一个空串；没有为该项添加全名。
- **Membership List** = 仅用于新组，组文件中列出的所有主体以及具有与该组相关的注册表帐户的所有主体。
- **Project List** = Yes（仅用于组）。

• 对于“帐户”项:
- **Account Expiration Date** = None。
- **Account-Valid Flag** = No。在设置口令之后，使用 **dcecp account modify** 命令，使这个标记更改为 **y**。
- **Client Flag** = Yes。
- **Duplicate Certificate Flag** = No。
- **Forwardable Certificate Flag** = Yes。
- **GECOS** = 与 **etc/passwd** 文件中主体的 GECOS 字段中的项相同的值。
- **Good Since Date** = 帐户创建的时间。
- **Home Directory** = 与 **/etc/passwd** 文件中主体的主目录项相同的值。
- **Login Shell** = 与 **/etc/passwd** 文件中主体的注册外壳项相同的值。
- **Maximum Certificate Lifetime** = 设置为注册表认证策略。
- **Maximum Certificate Renewable** = 设置为注册表认证策略。
- **Password** = 随机产生。注意，在用户认证可能发生之前，必须修改或重新设置随机产生的口令。
- **Password Date and Time Modified** = 设置为 **passwd_import** 运行的日期和时间。
- **Password-Valid Flag** = No。
- **Postdated Certificate Flag** = No。
- **Proxiable Certificate Flag** = No。
- **Renewable Certificate Flag** = Yes。
- **Server Flag** = Yes。
- **TGT Authentication Flag** = Yes。

注意，**passwd_import** 没有为它所创建的帐户设置可用的口令。在认证可能发生之前，必须使用 **dcecp account modify** 命令设置口令。还必须确保使帐户成为有效的。要完成这一任务，可在 dcep account modify 命令中指定 --acctvalid 选项，如下面的例子所示:

```
dcecp> account modify John_Hunter -acctvalid {yes}
```

如果没有完成这一任务，会导致主体无法使用 **dce_login**

passwd_import 命令语法

passwd_import 命令的语法为:

```
dceshared/bin/passwd_import [-h] [-c] -d pathname [-i] [-o org] \
     [-p  password] [-u  username] [-v]
```

其中:

-h 显示用法信息。

-c 以检查模式运行；处理显示冲突的命令，但不更改注册表。

-d *pathname*
至目录的路径，该目录包含要导入的口令和组文件。

-i 指定不冲突，表示同一身份的相同名称的字符串。

-o *org* 组织的名称，它分配给添加到注册表的所有主体。缺省值是名为 **none** 的组织。

-p *password*
帐户的口令，**passwd_import** 将以该帐户的特权运行。如果不使用 **-i** 选项，**passwd_import** 提示您解决名称冲突。

-u *username*
帐户的主体名，**passwd_import** 将以该帐户的特权运行。该帐户必须有权访问注册表并将主体、组、帐户和组织以及成员添加到组和组织。主体名和口令用于获得网络认证。如果没有提供它们，即使您已经执行了一次网络登录，**passwd_import** 也会提示您输入它们。

-v 以详细模式运行，生成所有活动的详细副本。

使用 passwd_import

要使用 **passwd_import**，安全性服务器必须是正在运行的。下列各部分描述如何使用 **passwd_import** 命令及其选项。

使用相同用户选项

-i 选项允许指定重复名称没有冲突，而是表示相同身份。 **passwd_import** 找到重复名称项时，它将它们视为同一个用户处理，并跳到下一个项。

使用检查模式

通过使用 **-c** 选项，首先以检查模式运行 **passwd_import**。在这种模式下，**passwd_import** 试图模拟处理的运行结果，显示当 **passwd_import** 在没有 **-c** 选项的情况下运行时遇到的冲突。

检查模式向您提供解决潜在冲突的数量和复杂程度的好方法。但是，检查模式不更改注册表。当运行不带 **-c** 选项的 **passwd_import** 命令并进行更改以解决冲突时，这些更改会依次创建在检查模式下不能迅速显示的进一步冲突。

如果在检查模式下遇到许多冲突，则在运行 **passwd_import** 之前，手工编辑注册表或 UNIX 组和口令文件来解决某些明显冲突是很有效的。

解决冲突

passwd_import 命令向您提示有关如何解决它所发现的冲突的指令。您有下列选项:

- 可以创建一个别名来解决 UNIX 标识冲突。这个操作为有冲突的注册表对象创建一个别名。这个别名被指定为与组或口令文件中的冲突项相同的名称。例如，如果注册表中存在项 **joe 555**，口令文件中存在项 **tim 555**，则该选项为 **joe 555** 创建别名 **tim**。
- 可以自动生成一个新的 UNIX 标识或明确地输入一个新标识来解决 UNIX 标识冲突。例如，如果注册表中的项 **joe 555** 与口令文件中的项 **tim 555** 之间有冲突，则可以为 **tim** 生成一个新的 UNIX 标识。

- 可以输入一个新名称来解决名称冲突。例如，如果注册表中的项 **joe 555** 和口令文件中的项 **joe 383** 之间有冲突，则可以为 **joe 383** 生成一个新名称。这个新名称将添加到注册表中。

另外，您可以选择忽略冲突并跳过该项。

回答提示

运行 **passwd_import** 时，可以提示您输入名称和编号（UNIX 标识）。名称可以包含除 @（at 符号）和 :（冒号）字符外的任何字符或数字，它们的长度不应该超过 1024 个字符。

如果用不正确的格式输入名称或编号，**passwd_import** 会忽略您的项并再次提示您输入。

样本 passwd_import 会话

本节显示简化的 **passwd_import** 会话。样本会话使用下列注册表组和口令项以及 UNIX 组文件和口令文件项。出于方便起见，注册表项以口令和组文件格式显示，尽管它们在注册表数据库中没有以这种格式存储。

注册表组和口令项

- 组项

```
wheel::0:
daemon::1:
none::2:
backup::3:user
locksmith::4:
login::5:
mail::6:bin
bin::7:root
server::8:
sys::9:root
staff::10:
sys_admin::11:user
sys_proj::12:
tgroup::35:
```

- 口令项

```
root:sq1RclUrrb1L6:0:10::/:
daemon:sq1RclUrrb1L6:1:2::/:
none:sq1RclUrrb1L6:2:2::/:
user:sq1RclUrrb1L6:3:2::/:
lp:sq1RclUrrb1L6:4:7::/:
sys_person:sq1RclUrrb1L6:5:2::/:
admin:sq1RclUrrb1L6:6:2::/:
uucp:sq1RclUrrb1L6:7:2::/usr/spool/uucppublic:
bin:sq1RclUrrb1L6:8:7::/:
```

UNIX 组和口令文件项

- 组文件项

```
system::0:root
other::1:
bin::2:root,bin,daemon
sys::3:root,bin,sys,adm
adm::4:root,adm,daemon
mail::6:root
rje::8:rje,shqer
```

```
daemon::12:root,daemon
tgroup::35:
diags::48:brown,smith,jones
cheetah::50:root,daemon
mkt_dev::52:roberts,anderson,hill
```

- 口令文件项

```
root::0:1:0000-Admin(0000):/:
daemon::1:1:0000-Admin(0000):/:
bin::2:2:0000-Admin(0000):/bin:
sys::3:3:0000-Admin(0000):/usr/src:
adm::4:4:0000-Admin(0000):/usr/adm:
uucp::5:5:0000-uucp(0000):/usr/lib/uucp:
rje::18:18:0000-rje(0000):/usr/rje:
trouble::70:1:trouble(0000):/usr/lib/trouble:
lp::71:2:0000-lp(0000):/usr/spool/lp:
setup::0:0:general system administration:/usr/admin:/bin/rsh
powerdown::0:0:general system administration:/usr/admin:/bin/rsh
sysadm::0:0:general system administration:/usr/admin:/bin/rsh
checkfsys::0:0:check diskette file system:/usr/admin:/bin/rsh
makefsys::0:0:make diskette file system:/usr/admin:/bin/rsh
mountfsys::0:0:mount diskette file system:/usr/admin:/bin/rsh
umountfsys::0:0:unmount diskette file system:/usr/admin:/bin/rsh
```

调用 passwd_import

在样本会话中，在外壳提示处输入下列 **passwd_import** 命令:

```
passwd_import -d sys5.3_tapes/adm -i
-v -u cell_admin
```

这个命令指定:

- 相同名称表示相同身份（**-i**）。
- UNIX 组和口令文件在 **sys5.3_tapes/adm** 目录中。
- 命令将不以检查模式运行（没有指定 **-c**）。
- 命令将以详细模式运行（指定 **-v**）。
- 其帐户应该用于认证的主体是 **cell_admin**。
- 因为没有使用 **-p** 选项，所以命令提示您输入 **cell_admin** 帐户的口令。

调用该命令之后，通过显示下列内容，系统准备 **passwd_import** 处理:

```
Preparing import files. (dce / sad)
Setting up registry information. (dce / sad)
Verifying that the necessary Organization exists. (dce / sad)
Creating group objects from group file. (dce / sad)
```

当 **passwd_import** 读取 UNIX 组和口令文件时，它向您通知所有冲突并提示它们的解决方案。

检查组文件

passwd_import 命令首先检查组文件的名称，然后检查 UNIX 标识冲突。当通过回答提示来解决冲突时，如果只使用 **passwd_import**，它会在注册表中创建组。

下列步骤显示如何处理 UNIX 标识组冲突:

1. **passwd_import** 命令首先找出两个 UNIX 标识之间的冲突，如上一节所示。组文件中的名称 **wheel** 和注册表中的名称 **system** 都有 UNIX 标识 **0**。**passwd_import** 命令向您提示如何解决冲突，如下所示:

```
CONFLICT: (wheel 0) - Import Group's UNIX id exists in registry.
          (dce / sad)
          (system 0) is the conflicting entry from the registry.
Do you wish to resolve the conflict (y) or skip this entry (n):
```

2. 如果输入一个 **n** 来跳过该项，则 **passwd_import** 继续处理。如果输入一个 **y** 来解决冲突，则 **passwd_import** 向您提示如何解决冲突。在下面的例子中，通过为注册表中 **system** 项创建别名 **wheel** 来解决冲突:

   ```
   Do you wish to resolve the conflict (y) or skip this entry (n): y
   Select one of: (a)lias, (g)enerate, (e)nter, (s)kip entry, (h)elp: a
   >> Adding Group entry for: wheel 0
   ```

   因为它正在以详细模式运行，所以 **passwd_import** 描述它正在执行的操作。每个操作描述都以 **>>**（重定向符号）开始。

   如果正在以检查模式运行 **passwd_import**，则不向您提示解决冲突。相反，会通知您有冲突并继续处理。显示的消息类似如下:

   ```
   CONFLICT: (wheel 0) - Import Group's UNIX id exists in registry.
           (dce / sad)
           (system 0) is the conflicting entry from the registry.
   Would need new UNIX id to resolve conflict. (dce / sad)
   ```

3. 如果 **passwd_import** 找不到必须解决的冲突，则当它处理组项时会显示它们以及它采取的操作，因为它正在以 verbose 模式运行。在下面的例子中，因为调用带 **-i** 选项的 **passwd_import**，所以没有提示您解决名称冲突。

   ```
   CONFLICT: (tgroup 35)
    - Group name exists in registry and UNIX ids match.
      (dce / sad)
   >> Import Group:  - Ignoring name conflict, as instructed
      (dce / sad)
   >> Adding Group entry for: diags 48
   >> Adding Group entry for: cheetah 50
   >> Adding Group entry for: mkt_dev 52
   ```

   当 **passwd_import** 继续处理 UNIX 组文件时，它找到两个其它 UNIX 标识冲突: UNIX 项 **adm 4** 和 **rje 8**，它们分别与注册表项 **locksmith 4** 和 **server 8** 冲突。

检查口令文件

然后，**passwd_import** 命令继续检查口令文件中的冲突。开始时，它显示下列内容:

```
Creating principal entries and accounts from password file.
(dce / sad)
```

如果处理项时没有冲突，则 **passwd_import** 在注册表中创建主体，将该主体添加到适当的组和组织中，并为该主体创建帐户。当它执行这些操作时，它显示下列内容:

```
>> Adding Principal entry for: rje
>> Adding account for rje none.
```

下面的例子显示当 **passwd_import** 发现冲突时显示的警告信息:

```
CONFLICT: (bin 2)
 - Principal name exists in registry and UNIX ids match.
   (dce / sad)
>> Import Principal:  - Ignoring name conflict, as instructed
       (dce / sad)
```

这条消息通知您，注册表中存在 **bin** 的帐户。即使 UNIX 口令文件项包含不同于注册表帐户的信息，这两个帐户也都保持不变。

因为在样本会话中使用的命令中指定了 **-i** 选项，所以忽略名称冲突。下面的例子显示当运行不带 **-i** 选项的 **passwd_import** 时发现的名称冲突提示:

```
CONFLICT: (daemon 1)
 - Principal name exists in registry and UNIX ids match.
   (dce / sad)
Do you wish to resolve the conflict (y) or skip this entry (n):
```

如果输入 **n**，则跳过该项并继续处理。如果输入 **y**，则 **passwd_import** 提示您为外来主体输入新名称，如下所示:

```
Enter new name for principal "daemon" "1":
```

将成员添加到组中

当 **passwd_import** 完成 UNIX 口令文件的处理时，它重新检查组文件，并将最近创建的主体添加到它在那里找到的任何附加组中。当它执行时，它显示下列内容:

```
Add memberships from imported group file. (dce / sad)
>> Add root as member of group with UNIX id: 0
>> Add root as member of group with UNIX id: 2
>> Add daemon as member of group with UNIX id: 2
```

完成处理

当 **passwd_import** 完成处理时，它显示下列内容:

```
Closing import files. (dce / sad)
Closing connection to registry. (dce / sad)
```

第40章 故障排除过程

本章包含调试安全性服务器的过程。仅当网络或硬件故障破坏注册表的操作时或当遇到无法用任何其它方法修补的问题时，才使用这些过程。这些过程告诉您如何执行以下操作:

- dce 守护进程核心位置的映射以及如何 symlink 到一个单独的文件系统
- 重新启动安全性服务器
- 当安全性服务器出现故障后，备份和恢复系统
- 以 locksmith 模式重新启动安全性服务器
- 从备份恢复副本
- 强制删除从属副本
- 访问用用户到用户协议注册的服务器
- 采用孤立的注册表对象，因为已经删除了它的所有者

在运行该过程之前，请先确保经由一个管理帐户登录。

DCE 守护进程核心位置的映射以及如何 Symlink 到一个单独的文件系统

这里是守护进程的映象以及它们转储核心的位置。

- secd - /opt/dcelocal/var/security/adm/secd
- dced - /opt/dcelocal/var/dced
- cdsd - /opt/dcelocal/var/directory/cds/adm/cdsd
- gdad - /opt/dcelocal/var/directory/cds/adm/gdad
- csdadv - /opt/dcelocal/var/adm/directory/cds/cdsadv
- cdsclerk - /opt/dcelocal/var/adm/directory/cds/cdsclerk
- dtsd - /opt/dcelocal/var/time/adm/dtsd
- pwd_strengthd - /opt/dcelocal/var/security/adm/pwd_strength
- idmsd - /opt/dcelocal/var/security/adm/idmsd
- emsd - /opt/dcelocal/var/sysmgmt/adm/emsd
- dceagtd - /opt/dcelocal/var/sysmgmt/adm/dceagtd
- audit - /opt/dcelocal/var/audit/adm
- nsid - /opt/dcelocal/var/directory/cds/adm/nsid
- session_manager - /tmp/sessmgr
- session_manager_control - /tmp/session_control

以下内容仅适用于 AIX:

- dceunixd - /opt/dcelocal/var/security/adm/dceunixd

以下内容仅适用于 Solaris:

- nss_dce_server - /opt/dcelocal/var/security/adm/nss

建议使用下列方案或相似方案:

使用 smit （smit 仅用于 AIX）创建名为 /cores/dce 的新文件系统。发出下列命令，使 /cores/dce 包含上面各个目录的个别目录。

- mkdir /cores/dce/secd
- mkdir /cores/dce/dced
- mkdir /cores/dce/cdsd
- mkdir /cores/dce/gdad
- mkdir /cores/dce/cdsadv
- mkdir /cores/dce/cdsclerk
- mkdir /cores/dce/dtsd
- mkdir /cores/dce/dceunixd

检查是否需要先将任何数据保存在任何这些目录中，如果需要的话，则保存数据，如日志文件或任何文件。

例如，发出

```
mv /opt/dcelocal/var/security/adm/secd/* /cores/dce/secd
```

对于任何含有要保存的数据的目录。

除去原始目录。

- rmdir /opt/dcelocal/var/security/adm/secd
- rmdir /opt/dcelocal/var/security/adm/dceunixd
- rmdir /opt/dcelocal/var/dced
- rmdir /opt/dcelocal/var/directory/cds/adm/gdad
- rmdir /opt/dcelocal/var/directory/cds/adm/cdsd
- rmdir /opt/dcelocal/var/adm/directory/cds/cdsadv
- rmdir /opt/dcelocal/var/adm/directory/cds/cdsclerk
- rmdir /opt/dcelocal/var/time/adm/dtsd

将核心位置 Symlink 到新的单独文件系统。

- ln -s /cores/dce/secd /opt/dcelocal/var/security/adm/secd/
- ln -s /cores/dce/dceunixd /opt/dcelocal/var/security/adm/dceunixd/
- ln -s /cores/dce/dced /opt/dcelocal/var/dced/
- ln -s /cores/dce/gdad /opt/dcelocal/var/directory/cds/adm/gdad
- ln -s /cores/dce/cdsd /opt/dcelocal/var/directory/cds/adm/cdsd
- ln -s /cores/dce/cdsadv /opt/dcelocal/var/adm/directory/cds/cdsadv
- ln -s /cores/dce/cdsclerk /opt/dcelocal/var/adm/directory/cds/cdsclerk
- ln -s /cores/dce/dtsd /opt/dcelocal/var/time/adm/dtsd

重新启动安全性服务器

要重新启动主或从属安全性服务器，可输入下列命令:

主:

```
start.dce sec_svr
```

副本:

```
start.dce sec_rep
```

为了方便起见，无论机器何时重新启动，将服务器设置为自动启动（使用 **config.dce** 上的 **-autostart yes** 选项来设置它）。

安全性服务器出现故障之后备份和恢复系统

注: 下列备份和恢复过程仅适用于遗留服务器，即，未迁移到 LDAP 迁移服务器或 LDAP 从属副本服务器的服务器。有关用于单元中已迁移到 LDAP 的所有安全性服务器的备份和恢复过程，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

如果安全性服务器出现灾难性故障，则应该使用下列过程。

备份注册表

注: 有关 “DCE 安全性注册表” 和 “LDAP 集成” 功能的该主题信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

1. 输入 registry disable 命令，将主副本设置为维护状态。下列命令将单元 giverny.com 中的主注册表设置为维护状态:

   ```
   dcecp> registry disable /.../giverny.com/subsys/dce/sec/master dcecp>
   ```

   如果将主副本设置为维护状态，可使主副本将它的数据库保存到磁盘并拒绝所有更新。

2. 选择步骤 2a 或步骤 2b 来备份注册表:

   注:

   用于备份的精确命令对于个人首选项很重要。但是，如果将数据库和主密钥文件写入同一个磁带，则将磁带存储在带受限访问权的锁定区域中。或者，可以将数据库和密钥文件写入单独磁带，并将每个磁带存储在不同位置中。

   a. 通过备份整个卷或 **/opt/dcelocal/var/security/rgy_data** 树（注册表）和 **/opt/dcelocal/var/security/.mkey** 文件（它包含用于加密注册表中所有键的主密钥），备份主注册表。注意，因为 **/opt/dcelocal/var/security/.mkey** 文件包含主密钥，恢复注册表数据库的备份是无用的，除非还恢复 **/opt/dcelocal/var/security/.mkey** 文件。

      使用 tar 备份下列各项:

      ```
      /opt/dcelocal
      /var/dce  (for AIX) and /opt/dcelocal/var  (for Solaris)
      /krb5
      /etc/dce  (for AIX) and /etc/dcelocal/etc  (for Solaris)
      ```

   b. 或者，下列 dceback 命令可以用来备份重要的安全性文件。

      要保存 “注册表” 文件:

      ```
      dceback dumpsecurity -destfile <filename>
      ```

      要保存 “客户机” 文件:

      ```
      dceback dumpmisc -destfile <filename>
      ```

3. 当备份完成时，使主副本脱离维护状态，如下所示:

```
dcecp> registry enable  /.../giverny.com/subsys/dce/sec/master dcecp>
```

安全性服务器继续接收更新。

注: 前面的例子为 **registry enable** 和 **registry disable** 命令提供了注册表主位置的名称。如果没有提供注册表的位置名称，则命令使用 **_s(sec_)** 变量中命名的位置。如果没有设置这个变量，则命令使用机器的缺省单元的主注册表。

恢复注册表

注: 有关 “DCE 安全性注册表” 和 “LDAP 集成” 功能的该主题信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

假定旧的安全性服务器不再启动，则使用下列过程:

1. 将新的机器安装到网络，为它指定与旧安全性服务器相同的 IP 地址和主机名。
2. 在该机器上安装 DCE，但不进行配置。
3. 选择步骤 3a 或步骤 3b 来恢复注册表:
   a. Untar 保存目录:

      ```
      /opt/dcelocal
      /var/dce  (for AIX) and /opt/dcelocal/var  (for Solaris)
      /krb5
      /etc/dce  (for AIX) and /etc/dcelocal/etc  (for Solaris)
      ```

   b. 或者，使用下列 dceback 命令来恢复通过 dceback 命令备份的数据:

      ```
      dceback restoresecurity -sourcefile <filename>
      dceback restoremisc -sourcefile <filename>
      ```

4. 重新启动系统。

以 Locksmith 模式重新启动主服务器

secd -locksmith 选项以 locksmith 模式启动 **secd**。这个选项只能在主副本上使用。在 locksmith 模式下，对 **secd** 指定的主体名成为 locksmith 主体。作为 locksmith 主体，可以修补恶意或意外更改，这些更改会使您不能以完全注册表访问特权登录。

当以 locksmith 模式启动安全性服务器时，**secd** 自动创建一个 locksmith 帐户，或者如果 locksmith 帐户存在，它让您为该帐户提供新口令。一旦安全性服务器运行，可以使用最近更改的口令（如果更改了它）登录到 locksmith 帐户，并使用您的正常凭证来访问注册表，以更改可能阻止您访问注册表的帐户或策略信息。

在 locksmith 模式下，所有具有有效帐户的主体都可以登录和操作具有正常访问检查的注册表。但是，locksmith 主体被授予对注册表的特殊访问权: 对经过认证的 locksmith 主体不执行访问检查。这意味着，作为 locksmith 主体，可以使用全部访问权操作注册表。

Locksmith 帐户的自动更改

如果在以 locksmith 模式启动安全性服务器时 locksmith 帐户存在，则安全性服务器检查确定的帐户和注册表策略信息，并进行第395页的表28 和第395页的表29 中显示的更改。这些更改确保，即使帐户或注册表策略被窜改，现在也可以登录到 locksmith 帐户。例如，如果入侵者将帐户使用期限注册表策略更改为 1 分钟，则 locksmith 帐户的有效期太短，无法使用它。所以，如果安全性服务器发现帐户使用期限注册表策略设置成

小于 locksmith 帐户所需的有效时间（至少 1 个小时），则它会将帐户使用期限策略更改为 locksmith 帐户的创建时间与当前时间之后的 1 小时之间的差。

表 28. 安全性服务器对 Locksmith 帐户进行的更改

如果安全性服务器发现...	它更改...
“口令有效标志”设置为 **no**	将“口令有效标志”更改为 **yes**
“帐户失效日期”设置为小于当前时间加 1 小时	将“帐户失效日期”更改为当前时间加 1 小时
“客户机标志”设置为 **no**	将“客户机标志”更改为 **yes**
“帐户有效标志”设置为 **no**	将“帐户有效标志”更改为 **yes**
“好的开始时间”设置为大于当前时间	将“好的开始时间”更改为当前时间
“口令失效日期”设置为小于当前时间加 1 小时	将“口令失效日期”更改为当前时间加 1 小时

表 29. 安全性服务器对注册表策略进行的更改

如果安全性服务器发现...	它更改...
“帐户生命跨度”设置为小于 locksmith 帐户创建日期与当前时间加 1 小时之间的差	将“帐户生命跨度”更改为当前时间加 1 小时减 locksmith 帐户创建日期
“口令失效日期”设置为大于上次更改口令的时间且小于当前时间加 1 小时	将“口令失效日期”更改为当前时间加 1 小时

以 Locksmith 模式启动安全性服务器

使用下面的 **secd** 命令格式，以 locksmith 模式启动安全性服务器:

dcelocal`/bin/secd [-locksm[ith] `*pname*` [-lockpw] [-rem[ote]]]`

其中:

-locksm[ith]
以 locksmith 模式启动安全性服务器。

pname 指定 locksmith 主体的名称。如果这个主体没有注册表帐户，则 **secd** 会创建一个帐户。

-lockpw
提示输入新的 locksmith 口令。这个选项允许您在不知道旧口令的情况下为 locksmith 帐户指定一个新口令。

-rem[ote]
允许 locksmith 主体远程登录。如果没有使用这个选项，则主体必须从将启动 **secd** 的本地机器登录。

以 Locksmith 模式重新启动安全性服务器

要以 locksmith 模式重新启动安全性服务器，可在运行主副本的节点上执行下列步骤。您必须有这个节点的 **root** 访问权。

1. 关闭安全性服务器。
   a. 如果无法用管理特权登录和访问 **dcecp** 来关闭服务器，则以 **root** 身份登录正在运行服务器的机器并杀死安全性服务器进程。

b. 如果可以用管理特权登录，则使用 **dcecp registry stop** 命令来关闭安全性服务器。使用这个命令时，必须将要停止的副本的全限定名作为一个变量提供。下面的样本命令停止名为 **slave_3** 的副本:

```
dcecp> registry stop  /.../giverny.com/subsys/dce/sec/slave_3
dcecp>
```

2. 以 locksmith 模式启动安全性服务器。下面的例子显示用 locksmith 帐户启动的安全性服务器，而该帐户是为名为 **master_admin** 的主体创建的。还提供了 **-remote** 选项，以允许 **master_admin** 从远程节点登录；否则， **master_admin** 必须从启动安全性服务器的节点登录。

   ```
   dcelocal/bin/secd -locksmith master_admin -remote
   ```

   如果 locksmith 帐户存在而您丢失了它的口令，则使用 **-lockpw** 选项会使 **secd** 提示您输入新的locksmith 口令并用输入的口令替换现有的口令。

安全性服务器在后台正常运行。当以 locksmith 模式启动安全性服务器时，它在前台运行，以便您可以回答提示。

一旦以 locksmith 模式启动安全性服务器，就可以使用 **dcecp registry modify** 命令来更改注册表，以便标准的特权帐户可以访问它。做完这些更改之后，应该执行下列操作:

1. 关闭正在以 locksmith 模式运行的安全性服务器。
2. 根据标准过程，重新启动安全性服务器。

恢复主副本

注: 有关 “DCE 安全性注册表” 和 “LDAP 集成” 功能的该主题信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

由于主副本的数据库遭损坏，要恢复主副本，可以使用下列任何一种方法:

- 使用 **dcecp registry designate** 命令，使从属副本成为主副本并在先前主副本的主机上创建一个从属副本。这种方法在下面部分中有描述。
- 从备份恢复主副本。这种方法在第361页的『第36章 执行例行维护』中有描述。

选择的方法取决于是主副本的备份数据库最新还是从属副本的数据库最新。

确定当前最新的数据库

注: 有关 “DCE 安全性注册表” 和 “LDAP 集成” 功能的该主题信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

要确定是主副本的数据库备份新还是从属副本的数据库备份新，可对副本运行 **dcecp registry show -replica** 命令。这个命令的输出列出最近的更新序号和更新日期和时间。将副本的最近更新序号和更新日期和时间与主副本的备份序号和日期和时间进行比较。如果是副本较新，则按照以下部分中的描述，使副本成为新的主副本。如果是主副本的备份较新，则按照第361页的『第36章 执行例行维护』中的描述，从备份恢复主副本。

将从属副本转换成主副本

注: 有关 “DCE 安全性注册表” 和 “LDAP 集成” 功能的该主题信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

本部分描述如何使用 **dcecp registry designate** 命令，将从属副本转换成主副本。要知道，因为 **registry designate -master** 命令会使数据丢失，所以不带 **-master** 选项的 **registry designate** 命令是指定不同主副本的首选方法。仅当对主副本的损坏不可恢复并且无法使用不带 **-master** 选项的 **registry designate** 命令时，才使用 **registry designate -master** 命令。

按照这些步骤，将主副本转换成从属副本:

1. 选择将成为新主副本的从属副本。
2. 发出下面的 **registry designate -master** 命令，将缺省主机更改为主注册表:

```
dcecp>  registry designate/.../musee.com/subsys/dce/sec/art -master
dcecp>
```

3. 使用 **registry show -replica** 命令，验证更改。
4. 使用标准的 UNIX命令，通过删除目录 *dcelocal*/**var/security/rgy_data** 和文件 *dcelocal*/**var/.mkey** 来删除旧主副本的数据库和 **.mkey** 文件。
5. 使用带 **-force** 选项的 **registry delete** 命令，从副本列表除去旧主副本。下面的例子从副本列表删除名为 **history** 的旧主副本:

```
dcecp> registry  delete /.../musee.com/subsys/dce/sec/history -force
dcecp>
```

恢复从属副本

注: 有关 “DCE 安全性注册表” 和 “LDAP 集成” 功能的该主题信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

因为没有备份从属副本，所以必须重新创建一个副本来恢复已毁坏的副本。要完成上述操作，使用下列过程:

1. 使用下列步骤取消配置 “安全性副本服务器”:
   a. 使用 **unconfig.dce** 命令来执行 **full** 或 **split** 取消配置。

      full 当 DCE 仍在 “安全性副本服务器” 设备上运行时使用以下命令:

      ```
      unconfig.dce sec_rep
      ```

      split 当 DCE 不在 “安全性副本服务器” 设备上运行时使用以下命令:

      - 在 “安全性副本服务器” 设备上运行:

        ```
        unconfig.dce -config_type local sec_rep
        ```

      - 在 DCE 单元中的其它机器上运行:

        ```
        unconfig.dce -config_type admin -dce_host <sec_rep_machine-dce_hostname> \
        -host_id <sec_rep_machine-host_id> sec_rep
        ```

      这些命令将完全清除 “安全性副本服务器”。

   b. 如果上述步骤不起作用，则使用下列命令:

      1) 使用标准的 UNIX 命令来手工删除副本的数据库文件和主密钥文件。要完成这一操作，删除下列位置中的所有文件:

- **/opt/dcelocal/var/security/rgy_data**
- **/opt/dcelocal/var/security/.mkey**

2) 使用 **set _s(sec)** 命令来绑定到主副本，然后使用 **dcecp registry delete -force** 命令，从主副本列表删除副本。以下两个命令显示如何绑定到主密钥文件，然后删除副本。

```
dcecp> set _s(sec) /.../musee.com/subsys/dce/sec/master
dcecp> registry delete /.../musee.com/subsys/dce/sec/art -force
dcecp>
```

3) 使用标准的 UNIX 命令，将文件 **/opt/dcelocal/etc/security/pe_site** 从运行主密钥文件的机器复制到将运行副本的机器。

4) 在副本机器上使用 **/etc/dce_config**（或提供者的等价物）来执行下列操作:
   a) 停止 DCE 守护进程 (stop.dce)
   b) 启动 DCE 守护进程 (start.dce)

2. 配置安全性服务器副本。该配置创建副本的数据库并启动 **secd** (config.dce)。
3. 当配置前面步骤中的副本时，为它指定一个名称。如果没有为这个副本提供与它先前名称相同的名称，则可能需要执行下列步骤，从 CDS 删除旧名称:
   a. 从 **/.:/subsys/dce/sec** 删除副本的服务器项名
   b. 从 CDS 组 **/.:/sec** 删除副本的名称

将主副本转换成从属副本

注: 有关 “DCE 安全性注册表” 和 “LDAP 集成” 功能的该主题信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

使用下列过程，将主副本转换成从属副本。仅当有多个主副本在网络或因特网上运行（这是例外情况）时，才使用这个过程。

1. 选择将成为从属副本的主副本。
2. 发出下面的 **registry designate -slave** 命令，将选择的主副本更改为从属副本:

```
dcecp> registry designate /.../dublin.com/subsys/dce/sec/lit -slave
dcecp>
```

3. 使用 **registry show -replica** 命令，验证更改。

强制删除从属副本

注: 有关 “DCE 安全性注册表” 和 “LDAP 集成” 功能的该主题信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

在本部分中描述的过程说明如何强制删除一个从属副本。仅当第371页的『第37章 处理网络重新配置』 中描述的普通删除方法失败时，才使用这种强制性方法。

要强制删除一个从属副本，可使用 **dcecp registry delete -force** 命令。这个命令从主副本的副本列表中删除从属副本。然后，主副本将删除请求传递给其它副本。由于这个操作从未与删除的副本通信，所以仅当副本终止并且无法重新启动时才使用 **-force** 选项。如果一个被强制删除的副本继续操作，则使用 **registry destroy** 来停止服务器并删除它的数据库。也可以简单地停止 **secd**（通过使用 **dcecp registry stop** 命令）并删除或重新命名它的数据库。

要强制删除一个注册表副本，发出带 **-force** 选项的 **registry delete** 命令，并将要删除的注册表作为一个变量提供。下列样本删除 **/.../giverny.com/subsys/dce/sec/lit_server_2** 的副本:

```
dcecp> registry delete /.../giverny.com/subsys/dce/sec/lit_server_2
-force
dcecp>
```

如果缺省副本不是主副本，**dcecp** 会自动绑定到主副本。

如果一个被强制删除的副本继续操作，则使用 **registry destroy** 来停止服务器并删除它的数据库。当使用 **registry destroy** 命令时，必须输入要停止的副本的名称。下面的例子显示 **registry destroy** 命令，用于删除 **/.../giverny.com/subsys/dce/sec/lit_server_2** 的副本。

```
dcecp> registry destroy /.../giverny.com/subsys/dce/sec/lit_server_2
dcecp>
```

或者，可以简单地停止 **secd**（通过使用 **dcecp registry stop** 命令）并通过删除或重新命名副本的数据库来毁坏它。

恢复重复主副本

注: 有关 “DCE 安全性注册表” 和 “LDAP 集成” 功能的该主题信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

本部分描述如何从一个十分罕见的问题恢复。不要使用在此描述的方法来解决问题，除非它是完全必要的。

有时候，要使其成为主副本的副本将有一个主序号，该序号小于（或等于）系统中的另一个主序号。当主副本检测到它的主序号小于系统中的另一个主序号时，它将自身标记为重复的主副本并且它的进程退出。每次启动主副本时，它将通知它已作为一个重复的主副本，并且它的进程将再次退出。

要强制这个重复的主副本成为主副本而不退出，可以用下面的格式重新启动带 **-master_seqno** 选项的 **secd** 进程:

secd -master_seqno *new_master_seqno*

其中，*new_master_seqno* 是要指定给副本的新主序号。使这个序号比系统中的最高主序号大 1。

使用 **dcecp registry dump** 命令来找到最高的主序号。

当前主副本出现故障时指定一个新的主副本

注: 有关 “DCE 安全性注册表” 和 “LDAP 集成” 功能的该主题信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

如果 “主安全性副本” 出现故障并不可用于扩展周期，则使用下列步骤来将一个现有的从属副本指定为新的主副本:

注: 完成这些步骤之后，先前的主副本不再配置在单元中。

1. 选择新的主副本位置。在您选择的主机上必须已经存在一个从属副本。
2. 登录到将成为新主副本的主机。
3. 编辑 **/opt/dcelocal/etc/security/pe_site** 文件，并除去旧主副本的所有项。然后，将 **TRY_PE_SITE** 环境变量的值设置为 **1**。
4. 以单元管理员身份登录 DCE。
5. 输入 **registry designate** 命令来设置新的主副本。当发出这个命令时，将要作为新主副本的副本的名称作为一个变量提供。在这个操作期间，不再接触到旧的主副本。这个命令可能要花几分钟完成。

   ```
   dcecp> registry designate /.../henry.com/subsys/dce/sec/cheyenne
   -master
   dcecp>
   ```

6. 输入 **registry show** 命令来验证新的主副本是否已成为主副本:

   ```
   dcecp> registry show /.../henry.com/subsys/dce/sec/cheyenne
   dcecp>
   ```

7. 输入 **registry delete** 命令，从副本列表除去旧的主副本:

   ```
   dcecp> registry delete /.../henry.com/subsys/dce/sec/bourbon -force
   dcecp>
   ```

8. 输入 **rpcgroup remove** 命令，从安全性 rpc 组 **/.:/sec** 和 **/.:/sec-v1** 删除旧的主副本:

   ```
   dcecp> rpcgroup remove /.:/sec -member /.../henry.com/subsys/dce/sec/bourbon
   dcecp>
   dcecp> rpcgroup remove /.:/sec-v1 -member /.../henry.com/subsys/dce/sec/bourbon
   dcecp>
   ```

9. 输入 **object delete** 命令，从 cds 名称空间删除旧主副本的 **cds** 对象:

   ```
   dcecp> object delete /.:/subsys/dce/sec/bourbon
   dcecp>
   ```

10. 输入 **acl modify** 命令，从副本列表 ACL 除去旧主副本的主机（取决于您所运行的 DCE 版本，它可能不在 ACL 中):

    ```
    dcecp> acl modify /.:/sec/replist
    -remove {user hosts/bourbon/self -i-m-I}
    dcecp>
    ```

11. 输入 **group remove** 命令，从安全性服务器组除去旧主副本的主机:

    ```
    dcecp> group remove /.:/subsys/dce/sec_servers -member /.:/hosts/bourbon/self
    dcecp>
    ```

12. 输入 **unconfig.dce** 命令，取消配置旧主副本的管理部分:

    ```
    $ unconfig.dce -config_type admin -dce_hostname bourbon sec_rep
    dcecp>
    ```

    注: 如果要取消配置旧主副本上的所有 DCE 组件，可以用 **all** 替换上述命令中的 **sec_rep**。

访问用用户到用户协议注册的服务器

如果服务器已经停止并重新启动，具有对已停止的旧服务器有未失效票券的客户机不能用已重新启动的新服务器执行经认证的 RPC。客户机必须等待当前票券失效或通过毁坏当前凭证 (kdestroy) 并执行新的 dce_login 来获得一个新票券。

采用注册表残留部分

尽管 **dcecp** 显示对象名并根据名称标识注册表对象，但 “DCE 安全性服务” 使用 UUID 来内部标识对象。当创建注册表对象时，“DCE 安全性服务” 自动设置对象名和它用来标识该对象的 UUID 之间的关联。当删除注册表对象时，删除注册表对象和标识该对象的 UUID 之间的关联。

因为已经删除主体或组，所以残留部分是由与主体或组不相关的 UUID 所拥有的对象。例如，如果从注册表删除主体，则还要删除用于外部标识主体的名称和用于内部标识主体的 UUID 之间的关联。已删除的主体所拥有的所有对象（文件、程序）现在由不再与主体相关的 UUID 内部拥有。如果其它主体、组或组织都无权访问该对象，则根本无法访问该对象并且它是一个*残留部分*。

要解决这处问题，可以使用带 **-uuid** 选项的 **dcecp principal create**、 **group create** 和 **org create** 命令来创建主体、组或组织，其 UUID 与拥有孤立对象的 UUID 相同，从而，采用孤立对象。

注: 当创建新的注册表对象，无法指定与该对象相关的 UUID; 所以，不能简单地添加一个名称相同的新注册表对象来采用残留部分。

-uuid 选项创建主体、组或组织，并允许指定应与之相关的 UUID，以代替自动指定给它。除了创建方式之外，通过这些命令创建的主体、组或组织与任何其它主体、组或组织无差别。下面的例子显示如何使用这个选项创建主体、组或组织，以采用孤立的注册表对象。

要创建与拥有孤立对象的 UUID 相关的主体，可使用下面的命令:

```
principal create name -uuid uuid [-fullname fullname] \
    [-quota object_creation_quota] [-uid UNIX_number]
```

要创建与拥有孤立对象的 UUID 相关的组，可使用下面的命令:

```
group create name -uuid uuid [-fullname string] \
    [-inprojlist [yes| no]] [-gid UNIX_number]
```

要创建与拥有孤立对象的 UUID 相关的组织，可使用下面的命令:

```
organization create name -uuid uuid \
[-fullname string] [-orgid UNIX_number]
```

其中:

name 主体、组或组织的主名称。

uuid 要指定给主体、组或组织的 UUID 号。这个 UUID 应该是拥有孤立对象的那个 UUID（即，与已删除的注册表对象相关的那个 UUID）。 UUID 在 RPC 打印串格式中被指定为 8 位十六进制，一个连字符; 4 位十六进制，一个连字符; 4 位十六进制，一个连字符; 4 位十六进制，一个连字符; 和 12 位十六进制。格式如下:

nnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnnnnnn

string 主体、组或组织的全名。

UNIX_number

对于单元主体，仅与名称相关的 UNIX 号。如果没有输入这个选项，则会提供

下一个连续的 UNIX 号。对于除单元外的所有主体，UNIX 号是从嵌于主体的 UUID 中的信息抽取的并且不能在此指定。

object_creation_quota
对于主体，仅主体的对象创建限额。如果没有输入这个选项，则缺省值是 **unlimited**。

-inprojlist
对于组，仅 **yes** 关闭项目列表内含物，以使项目列表中不包含组。如果输入 **no**，则项目列表包含组。

注: 在当前的 DCE 实现中，UNIX 号嵌入 UUID 中。如果尝试创建一个组或组织来采用孤立对象并且失败，这可能因为嵌入的 UNIX 号是无效的，因为它不在为单元设置的有效 UNIX 号范围（作为注册表特性）内。如果出现这种情况，必须重新设置有效 UNIX 号的范围，以包括嵌入 UUID 中的 UNIX 号，然后，再次尝试采用该对象。请参阅第353页的『第35章 维护策略和特性』，以获得有关设置 UNIX 号的有效范围信息。

第41章 AIX/DCE 安全性集成（仅 AIX）

在这个发行版中，AIX 基本操作安全性服务已经与 DCE 安全性服务一起集成。这种集成是为了为一般的最终用户提供单一的系统映象,而不是分开的本地 AIX 系统和远程 DCE 系统映象。如“更多详细信息--安全性集成”中所解释的，存在一些集成限制；然而，大多数用户可以通过 AIX 命令（如 **login** 和 **su**）获得 DCE 凭证，可以通过 AIX **passwd** 命令更改其 DCE 口令，并可以通过标准的 AIX **libc.a** 例程 **getpwnam( )**、**getpwuid( )**、**getgrnam( )** 和 **getgrgid( )** 从 DCE 注册表检索 AIX 类型信息。目前，更一般的例程 **getpwent( )** 和 **getgrent( )** 还不支持 DCE。

更多详细信息--安全性集成

强烈建议：DCE 注册表中的用户和组与单独的系统 **/etc/passwd** 和 **/etc/group** 文件保持紧密同步。可以在配置了初始单元之后用 **passwd_export** 和 **passwd_import** 实用程序实现同步。任何没有在单元注册表和本地文件中同步的用户都无法在以后体验到集成的完整好处。另一方面，这种灵活的集成模式支持漫游用户（在 DCE 注册表中定义、但不在本地系统中的用户）。如果配置了机器以允许它，则那些漫游用户可以登录到系统，并基于注册表中与 UNIX 相关的信息获得 DCE 凭证和本地访问权。

非集成的 DCE 安全性命令 **dce_login**（用于登录到 DCE）和 **dcecp** 程序（用于更改口令和查询注册表）仍然可用。当然，必须具有本地访问权才能使用这些命令。

本地管理员必须设置系统，以启用 AIX/DCE 安全性集成。这名管理员应该以本地用户 **root** 的身份执行本节中的步骤。应该已经将系统配置成 DCE 客户机机器。

启用 DCE 访问

/etc/security/user 文件定义特定用户（具有节的用户）和其它用户（使用缺省节的用户）的属性。为了支持 AIX/DCE 安全性集成，定义了两个新的用户属性 **-- SYSTEM** 和 **registry**。使用 **SYSTEM** 属性来允许或拒绝特定用户或整个系统的 DCE 访问。使用 **registry** 属性来强制本地或到 DCE 的口令操作。

需要了解的一点是：只对与域相关的名称应用这些和其它用户属性。那意味着，以 **jane**、**/.:/jane** 或 **/.../this_cell/jane** 登录到系统的漫游 DCE 用户受可以在 **/etc/security/user** 中本地用户 **jane** 的节中表示的属性的影响。

SYSTEM 属性

SYSTEM 属性描述用来对系统认证用户的方法。这个属性的有效值是由以下标志组成的布尔表达式字符串:

files 只通过本地 **/etc/passwd** 文件认证。

compat
通过本地文件和／或 NIS 数据库认证。

DCE 通过 DCE 注册表认证。

SYSTEM 属性值可以是由上面的方法（标志）和 **AND** 和 **OR** 布尔运算符组成的复杂表达式。而且，**SYSTEM** 表达式可以指定方法所得出的不同结果；语法是 `method`

[result]。有效结果是 UNAVAIL（认证服务不可用）、NOTFOUND（在数据库中没找到用户）、 FAILURE （由于不同的、未指定的原因，认证失败）或 SUCCESS（方法中所固有，无结果）。

缺省节用 SYSTEM = compat 创建。然后，本地管理员必须更改这个节，或要求采用另一种认证模式的任何用户的特定用户节。本地用户 root 的属性应该值总是 SYSTEM = compat（以及 registry = files）；在那种方式下，超级用户就不依赖于非本地认证模式。

有关 **SYSTEM** 用户属性及其合法值的进一步讨论，请参阅 AIX 文档。

注册表属性

registry 属性定义了管理用户口令的数据库。这个属性确定在哪里进行口令查询和更改。该属性的有效值是:

files 在本地文件中管理用户口令。

NIS 在 NIS 数据库中管理用户口令。

DCE 在 DCE 注册表中管理用户口令。

registry 没有缺省值，并且，如以后将要讲的，管理员可以选择不定义缺省值。请注意，应该始终将本地用户 **root** 定义成 registry = files（以及 SYSTEM = compat）。以那种方式，超级用户口令操作就不依赖于非本地机制。

有关 **registry** 用户属性及其合法值的进一步讨论，请参阅 AIX 文档。

保护本地资源

本地系统管理员必须保护所有本地资源。如前面所述，我们强烈建议 DCE 注册表和本地 **/etc/passwd** 和 **/etc/group** 文件尽可能保持紧密同步。然而，在某些情况下，可能不希望保持本地用户和注册表的同步（本地用户 **root** 就是个好例子）。在那些情况下，本地管理员必须保护非同步的名称和／或 ID。

本地管理员应该采取措施来保护已知与 DCE 注册表不同步的、以及可能在本地系统上存有数据的所有名称（用户或组名称）和 ID（UID 或 GID）。例如，如果本地机器上存在用户 **joe** 并且在 DCE 注册表中存在一个不同的用户 **joe**，则本地管理员应该确保: DCE **joe** 无法访问该机器。

以下各节描述保护本地资源的不同方法。

用 /etc/security/user 文件保护本地资源

对于已知不与注册表中的同名用户同步的任何用户，应该在 **/etc/security/user** 中用其 **SYSTEM** 和 **registry** 属性将其定义成本地（或 NIS）用户。

要防止任何漫游用户获得对本地系统的访问权，缺省节应该从 **SYSTEM** 属性排除 DCE:

```
default:          SYSTEM = compat
                  registry = files
```

与 DCE 注册表同步的本地用户仍然可以通过在其单独的节中将 DCE 指定成认证方法来在登录时获取 DCE 凭证。

用 passwd_override 和 group_override 文件保护本地资源

标准的 DCE **passwd_override** 功能对集成的认证有效。也就是说，本地管理员可以通过在 **/opt/dcelocal/etc/passwd_override** 文件中放入适当的项，以根据用户名、UID 或 GID 来防止 DCE 认证的继续进行。例如，以下 **passwd_override** 项防止用户 **joe** 进行认证:

```
joe:OMIT:::::
```

类似的 UID 或 GID 的 **passwd_override** 项可以防止具有指定的标识的用户进行 DCE 认证。

对于基本 DCE， **/opt/dcelocal/etc/group_override** 文件用于修改在 **passwd_export** 操作期间创建的组项。在具有 AIX/DCE 集成的安全性特性的系统上，可以使用 **group_override** 文件来防止 DCE 用户进行本地访问。要这样做，在 **group_override** 项中指定组名称和 OMIT。然后，那个组中的所有用户成员被拒绝进行本地访问。例如，项

```
temps:OMIT::
```

防止属于 **temps** 组的任何 DCE 用户登录到本地系统。**temps** 是用户的组集中的主组还是任意组并不重要。 **group_override** 文件实施的访问拒绝只对集成的 BOS 功能有效（而不是象 **dce_login** 这样的仅 DCE 功能），并且只对组名称实施拒绝。要基于 GID 来拒绝访问，使用 **passwd_override** 文件机制。

在本发行版中，**dced** 守护程序实现覆盖功能。因为覆盖功能是保护本地资源不可或缺的一部分，所以，DCE 认证被设计成: 如果 **dced** 守护程序不可用，它就失败。本地管理员必须确保: 如果必要的话，还可以通过其它认证方法来进行本地访问。有关 **passwd_override** 和 **group_override** 文件的完整讨论，请参阅 DCE for AIX Administration Command Reference。

注: 因为 dced 在瘦客户机上不可用，所以，**passwd_override** 和 **group_override** 功能也不可以在瘦客户机配置上使用。

注: 由于性能方面的原因，最好用 **/etc/security/user** 文件（如果可能的话）、而不是 DCE 覆盖文件来保护本地资源。

按每用户配置 DCE 访问

以下各节提供了有关对每个用户配置访问权限的特定详细信息。

配置和保护本地 Root 用户

本地超级用户的认证不应该依赖于任何非本地机制。因此， **/etc/security/user** 应该始终包含一个具有以下项的 root 节:

```
root:         SYSTEM = compat
              registry = files
```

以这种方式，所有认证和口令操作都导向本地文件。这还防止 DCE 用户 **root** 登录到这个系统上的 DCE（除非 DCE 用户 **root** 的口令与与 root 口令偶然匹配）。

还可以选择使用 **passwd_override** 实用程序来防止其它具有 UID **0** 的 DCE 用户登录到系统。只有在系统上启用了漫游 DCE 用户认证，并且单元管理员已经创建了 DCE 用户 **root** 的别名时，这才有必要。可以用 **dcecp** 查看 DCE 注册表中的帐户来确定别名是否存在。

配置和保护仅本地用户

仅本地用户是在本地系统上定义、但是不与 DCE 注册表中的任何主体同步的用户。因此，一些 DCE 用户的名称或标识可能与本地用户冲突。这样的用户应该定义 **SYSTEM** 和 **registry** 属性的 **/etc/security/user** 中具有特定的节，如下所示:

```
joe:
        SYSTEM = compat
        registry = files
```

注: 如果合适的话，可以将 **SYSTEM** 设置成 **files**，并将 **registry** 设置成 **NIS**。目的是明确地从属性中排除 DCE。

因此，本地用户 **joe** 不依赖 DCE 进行认证或口令操作。这还防止任何 DCE 用户 **joe** 登录到系统上的 DCE。

还可以通过创建 UID 或 GID 的 **passwd_override** 项或创建组的 **group_override** 项来保护仅本地用户的系统资源。

配置同步的用户

在正确集成的单元中，大多数用户都在 DCE 注册表中定义和管理，并且任何存在于本地的用户都应该与 DCE 用户同步。除了口令之外，可以使用 **passwd_export** 实用程序来将所有本地用户信息与 DCE 注册表同步。然后，应该使用 AIX **passwd** 命令或 **pwdadm** 管理命令来同步口令。

任何已同步的用户可以使用 AIX BOS 命令，并将它们导向 DCE 注册表。要这样做，**/etc/security/user** 中用户的节（特定节或缺省）应该有将 DCE 定义成首先要尝试的认证方法的 **SYSTEM** 属性，如下例所示:

```
      SYSTEM = "DCE OR (DCE [UNAVAIL]  AND compat)"
或
      SYSTEM = "DCE OR (DCE FAILURE]  AND compat)"
或
      SYSTEM = "DCE AND compat"
```

注: 对于已同步的用户，不设置注册表属性。

在以上示例中，DCE 是首先尝试的认证方法。如果通过了认证，则用户被授予访问权；通过 DCE 获得所有 UNIX 类型的信息（UID、GID、主目录、登录外壳）和 DCE 凭证。在第一个示例中，如果 DCE 由于不可用而失败，则尝试本地认证。如果 DCE 认证由于任何原因失败，则第二个示例尝试本地认证。在允许用户访问系统之前，第三个示例要求 DCE 和本地认证成功。

用户的 **AUTHSTATE** 环境变量被设置成第一个成功的经认证的方法，并且后续操作被导向由 **AUTHSTATE** 定义的注册表（请参阅本节中的“访问方法标识”）。

不要设置已同步用户的 **registry** 属性；**AUTHSTATE** 环境变量用于指出口令操作被导向的正确的注册表。换句话说，如果 **AUTHSTATE** 是 **DCE**，则 **registry** 是 **DCE**，

如果 **AUTHSTATE** 是 **files** 或 **compat**，则 **registry** 是 **compat**。如果将注册表属性显式地设置成与用户的 **AUTHSTATE** 冲突的注册表，则口令操作可能会失败。

支持漫游 DCE 用户

漫游 DCE 用户是在 DCE 注册表中定义、但不在本地机器中定义的用户。可以通过设置缺省 **SYSTEM** 属性来允许漫游用户从任何机器登录到 DCE，如下所示:

```
default:
        SYSTEM = "DCE OR compat"
```

任何 DCE 用户都可以通过提供其 DCE 名称和口令来从任何一台这样配置的机器登录到 DCE。当然，如果存在具有冲突的名称或标识的本地用户，并且通过其自己在 **passwd_override** 或 **group_override** 文件中的节项保护起来，则拒绝漫游用户访问系统。

访问方法标识

认证之后，在用户环境中设置 **AUTHSTATE** 变量。这个变量定义由用户实现以访问系统的认证机制，并可以具有 **files**、**NIS** 或 **DCE** 值。**AUTHSTATE** 变量确定被访问以提供用户后续操作的第一个数据库。然而，如果定义了用户的 **registry** 属性，则将口令操作导向由 **registry**、而不是用户的 **AUTHSTATE** 定义的数据库。本地管理员应该确保: 用户的 **registry** 属性不与由 **SYSTEM** 属性间接确定的用户的 **AUTHSTATE** 冲突。一种方法是只定义仅本地用户的、而不定义其它用户的 **registry** 属性。应该将仅本地用户（例如 **root**）定义成 `registry = files`。

更改口令

口令操作被导向由 **registry** 用户属性（或者，如果注册表属性定义不存在，则由 **AUTHSTATE** 环境变量定义的 **registry**）定义的注册表。不将口令同时导向本地和 DCE 注册表。通过 **dcecp** 更改仅 DCE 用户的口令，但是可以通过 AIX **passwd** 命令分两步更改已同步用户的口令（DCE 和本地）:

```
$ AUTHSTATE=DCE passwd
$ AUTHSTATE=compat passwd
```

用户应该在更改 **AUTHSTATE** 值之前回显它，并在更改完成之后将其设置回初始值。

注: 必须同步化同步用户的口令。如果没有同步化，则 DCE 或本地认证将失败。另外，如果一个用户在多台机器上存在，则必须同步化所有机器上的本地口令。

故障排除

以下描述了一些可能出现的情况，需引起管理员注意。任何与 BOS 命令相关的问题还可能负面影响系统管理员。

dceunixd 将不启动，发出消息 (dceunixd) 确保存在有效的机器凭证，并且没有设置 **KRB5CCNAME** 环境变量。

原因: 如果现有凭证（由 **KRB5CCNAME** 环境变量确定）不存在，则可能会发生这种情况。

操作: 如果在环境中设置了 **KRB5CCNAME**，则取消它的设置。还要验证机器凭证 **(/opt/dcelocal/var/security/cred/dcecred_ffffffff)** 是否存在。您可能需要启动（或重新启动） **dced** 来创建新的机器凭证。

dceunixd 将不启动，发出消息 (dceunixd) 检测到另一个 dceunixd 实例、pid XXXXX（其中，**XXXXX 代表进程 ID**）。

原因: 如果 **dceunixd** 已经在机器上运行，则可能会发生这种情况。

操作: 确保带有所述标识的 **dceunixd** 进程正在运行。如果是，则无法启动另一个 **dceunixd**。如果没有这样的进程存在，则除去以下文件:

```
/opt/dcelocal/var/security/adm/dceunixd.skt
/opt/dcelocal/var/security/adm/dceunixd.pid
```

然后重新调用 **dceunixd**。

dceunixd 进程过度增长。

原因: 当大量认证请求在机器上发生时，可能会发生这种情况。每一个 DCE 认证将导致认证进程的增长，因为 **dceunixd** 代表很多客户机认证，它可以增长得很大（取决于系统活动）。

操作: 如果因为 **dceunixd** 正在消耗大量内存而导致系统性能降低，则停止、然后再重新启动守护程序。应该考虑在 **dceunixd** 停止处运行 **cron** 作业，然后以常规方式重新启动。

AIX 命令将不运行，发出消息 无法装入安全性方法 "DCE"。

原因: 如果没有为 AIX/DCE 安全性集成正确定义机器，则可能发生这种情况。

操作: 确保 **dceunixd** 守护程序正在运行，并且在机器上安装了模块 **/usr/lib/security/DCE**。它应该具有以下所有权和许可权:

```
-rwxr-xr-x  1 root    system
```

有关其它配置信息，请参阅“步骤 --安全性集成”。

ls -l 命令将 UNIX 标识映射成错误的用户或组名，或者将标识映射成名称失败。

原因: 如果 **AUTHSTATE** 环境变量指向意外的数据库，则可能出现这种情况。

操作: 检查 **AUTHSTATE** 环境变量。所有 **getpw*** 和 **getgr*** 操作（除了 **getpwent( )** 和 **getgrent( )**）都首先导向由 **AUTHSTATE** 指出的数据库。也就是说，如果 AUTHSTATE = DCE，则查询将导向 DCE 注册表。如果没有在 DCE 中找到名称或 ID，则查询本地或 NIS 数据库。或者，如果 AUTHSTATE = compat，则将在 DCE 之前查询本地数据库。因此，查询操作的行为可以影响标识到名称映射命令（**ls -l** 或 **id**）的输出。只有在本地用户标识与 DCE 用户标识发生冲突（未同步的用户）的系统上才会出现错误映射或不明确映射。

认证（例如 su、login、telnet、ftp）失败。

原因: 取决于系统和用户配置，可能有多种原因导致这种情况的出现。

操作:

1. 检查用户的 **SYSTEM** 属性。**/etc/security/user** 中的用户（或缺省）节必须指定正确的认证机制。对于已同步的用户（在本地和 DCE 中都定义了的用户），应该定义 **SYSTEM** 属性，以便在认证失败时进行仅本地认证。

2. 漫游用户应该在不同的机器上尝试认证。由于本地系统上有显式的保护措施，所以可能会被拒绝访问。
3. 检查 **dced** 是否在运行。如果 **dced** 不可用，则 DCE 认证失败。

没找到注册表对象。

原因: 命令实际上可能已经成功完成。

例如，考虑这样一台机器，该机器被配置成将安全性操作缺省导向 **DCE OR compat**（也就是说， **/etc/security/user** 缺省节中的 **SYSTEM** 属性等于 **DCE OR compat**）。如果试图在那台机器上 **su** 到仅本地用户，则在登录期间查询 DCE 注册表（因为那是在 **SYSTEM** 属性中指定的第一个机制）。从 DCE 返回 “没找到” 消息，并且在继续到下一个认证方法 (**compat**)（它应该成功）之前，**su** 进程显示它。

操作: 对于用户，发出 **id** 命令来确认是否发生了认证。对于系统管理员，在 **/etc/security/user** 文件中查看，以确定特定用户正在使用的认证方法。

用户应该知道: 可以将给定系统配置成将安全性操作首先导向 DCE，然后再导向本地系统。接下来，仅本地用户上的操作（例如 **su** 和 **login**）可能导致在它实际在本地成功完成之前就显示一条 DCE 错误消息。

发生集成登录失败。

原因: 如果 DCE 响应时间太慢，则可能会在集成登录期间失败。在集成登录过程中，将对 DCE 注册表发出很多不同的查询，如果这些查询中的任意一个不够及时，则登录过程将超时并失败。

象网络通信量、WAN 的速度和从 DCE 注册表返回的数据大小等变量确定失败的阈值。这种变量的组合还可以导致其它 DCE 性能问题。

操作: 类似的集成登录故障可以通过配置注册表加以减缓，将注册表配置成没有组有多于大约 1000 个用户，并且没有用户是多于大约 1000 个组的成员。这样的配置限制了可能要在登录尝试时处理的数据量。

因为 DCE 故障可以影响集成安全性操作，所以，应该确保: 系统应该在必要时退回仅本地操作。

步骤--安全性集成

执行以下步骤来配置集成安全性操作的系统:

1. 确保在机器上安装了模块 **/usr/lib/security/DCE**。
2. 编辑 **/etc/security/login.cfg** 文件以包括以下行:

```
DCE:
        program = /usr/lib/security/DCE
```

这定义 DCE 到系统的认证方法。

3. 确保守护程序 **dceunixd** 运行在机器上。这个守护程序代表 BOS 命令与 DCE 服务器 **secd** 和 **dced** 通信。有关详细信息，请参阅 DCE for AIX Administration Command Reference 中的 **dceunixd** 参考页面。

4. 编辑 **/etc/security/user** 节文件，以允许和／或拒绝用户的 DCE stanza f访问。有关编辑该文件的指导信息，请参阅“更多详细信息 --安全性集成”下的“启用 DCE 访问”。
5. 创建或编辑 **/opt/dcelocal/etc/passwd_override** 和 **/opt/dcelocal/etc/group_override** 文件，以显式地防止特定用户的 DCE 访问。有关编辑该文件的指导信息，请参阅“更多详细信息 --安全性集成”下的“保护本地资源”。

限制--安全性集成

在这个发行版中，对集成功能和利用该功能的用户有一定的限制。这些限制是:

- 任何通过 AIX 基本命令（如 **login**、 **su** 或 **telnet**）获得的 DCE 凭证都维护 DCE 语义。也就是说，它们的失效时间由 DCE 注册表确定，并且不能在本地登录会话期间继续存在。当运行 **rmxcred** 时必须要小心，在登录和许多活动的窗口中建立的凭证可以被删除，并且必须执行完整的登录以重新获得一致（跨窗口）的 DCE 凭证。
- 如果在使用 DFS，则应该在启动 **dceunixd** 之前在您的机器上配置和运行 DFS 客户机。任何促使集成的 **login** 或 **dce_login** 在 DFS 运行之前去获得 DCE 凭证的用户将没有对 DFS 文件的授权访问权。
- 任何与单元相关名称长于八个字符的 DCE 用户都无法利用集成特性。八个字符的限制是 UNIX 施加的限制。
- 任何与单元相关名称中包含斜杠 (/) 的 DCE 用户都无法利用集成特性。因此，DCE 用户 **mary** 可以利用集成特性,而 DCE 用户 **writers/mary** 则不能.
- 只支持本地用户；外部主体无法使用集成操作。外部主体仍然可以使用 **dce_login** 和 **dcecp** 进行单元间访问。
- 某些由 AIX 生成的本地用户数据可能会在允许漫游 DCE 用户登录以及本地用户没有被足够保护起来的系统上被错误表示。该数据包括上一次登录时间和系统审计跟踪中的信息。如果具有同样名称的 DCE 用户访问系统，则本地用户的信息将受到影响。例如，考虑这样的情形: 本地系统定义了一个用户 **joe**，并且该用户与 DCE 用户 **joe** 不同。假设本地系统和 DCE 注册表上的 UID 或 GID 之间没有冲突。本地管理员没有能够成功地覆盖 DCE 用户 **joe** 的登录（通过 **passwd_override** 文件），并已经通过将 **/etc/security/user** 中的缺省 **SYSTEM** 值设置成 `compat OR (compat [FAILURE] AND DCE)` 而启用了漫游用户登录。

  如果 DCE 用户 **joe** 登录到这个系统，则他将获得访问权，并具有从 DCE 注册表获得的 UID 和 GID。因此，DCE **joe** 无法访问本地 **joe** 的文件，因为它们由本地 **joe** 的 UID 和 GID 所有。然而，本地系统仍然知道用户是 **joe**，并且在某些场合下，它无法区分自己属于哪个域（DCE 或本地）。因此，将记录的数据归为 **joe**，而不管是 DCE **joe** 还是本地 **joe** 生成了数据。

  管理员应该通过将所有本地用户尽可能地与 DCE 注册表同步，并对未经同步的用户加以足够的保护（通过对冲突名称应用本地覆盖和正确配置 **/etc/security/user** 节）来防止这种问题的发生。
- 本地 **root** 用户无法 **su** 到 DCE 用户并获得 DCE 凭证。这是因为本地 **root** 用户没有被提示输入 DCE 登录所必需的口令。事实上，任何不提示输入口令的命令（例如，当 **.rhosts** 文件存在时的 **rcp**、**rsh** 或 **rlogin**） 都不能用于获得 DCE 凭证。获得 DCE 凭证的 BOS 命令是支持 DCE 的，因为它们使用 DCE 注册表来获得用户和口令信息。它们不使用底层的 Kerberos 认证协议。

- AIX **passwd** 命令只更改那些其 **pwd_val_type** 扩展注册表属性指定了由用户选择口令的 DCE 用户的 DCE 口令。它无法各个那些其 **pwd_val_type** ERA 指定了由系统生成口令的 DCE 用户的口令。
- 本地 DCE 注册表不足以通过 DCE 认证。如果 DCE 安全性服务器不可用，并且 **SYSTEM** 属性将 DCE 定义成必需的认证方法，则用户无法访问系统。
- 口令操作被导向 DCE 注册表或本地文件，而不是同时导向二者。因此，各个在 DCE 和本地系统中都存在的用户（作为同步的用户）的口令的更改是一个分为两步的过程。可以使用 AIX **passwd** 命令来首先更改一个注册表中的口令，然后再更改另一个。例如:

```
$ AUTHSTATE=DCE passwd
$ AUTHSTATE=compat passwd
```

某些依赖于 **/etc/passwd** 或 **/etc/group** 信息的 AIX BOS 命令不支持 DCE。这些命令包括 **finger**、**quot**、**lsuser** 和某些管理命令（例如 **mkuser**、**rmuser**、**mkgroup** 和 **rmgroup**）。

示例--安全性集成

- 用户 **joe**（其本地 **/etc/passwd** 信息与 DCE 注册表同步）可以配置一个节，以便在登录时获得 DCE 凭证和本地访问权。他的节类似于:

```
joe:
     SYSTEM = "(DCE AND compat) OR  (DCE[UNAVAIL] AND compat)"
```

这表明: **joe** 必须通过 DCE 和 **compat** 认证，否则他必定由于 DCE 不可用而无法通过 DCE 认证，并且将通过 **compat** 认证并成功登录到系统。

- 另一个示例是，漫游用户 **bob** 可以尝试登录到缺省节包含以下信息的系统（他没有在该系统上定义）:

```
default:
       SYSTEM = "DCE OR (DCE [UNAVAIL] AND compat)"
```

这表明: 缺省用户必定通过 DCE 认证或由于 DCE 不可用而无法通过 DCE 认证，并且将通过 **compat** 认证并获得对系统的访问权。当然，如果用户是漫游用户并且在 **/etc/passwd** 中没有本地项，则该用户无法通过 **compat** 并且在 DCE 不可用的情况下将被拒绝访问权。

- 如果知道本地用户不与在注册表中具有同样名称的用户同步，则本地用户 **joe** 的节将包括以下项:

```
 joe:
      SYSTEM = compat
      registry = files
```

NIS 用户 **jane** 的节将包括以下项:

```
jane:
       SYSTEM = compat
       registry = NIS
```

目的是明确地从属性中排除 DCE。如果 **joe** 或 **jane** 漫游到机器，则他们可以登录的唯一方法是提供使 **compat**（本地或 NIS）认证成功的口令。

- 如果用户 **jane** 通过 DCE 访问本地系统，则她的 **AUTHSTATE** 变量被设置成 **DCE**。如果她接下来对某些文件执行 **ls -l** 命令，则将查询 DCE 注册表以将 UID 和 GID 映射成名称。如果在 DCE 注册表中不存在 ID，，则 **ID query** 操作将回退到

compat，并且将查询本地 **/etc/passwd** 和 **/etc/group** 文件。用户应该清楚：根据可能被查询的数据库和正在被访问的文件系统（DFS 或本地）的不同，可能会错误地映射 ID。

单元间考虑--安全性集成

AIX/DCE 安全性集成可以用来允许从外部单元的 DCE 用户登录并获得其外部单元的 DCE 凭证。这通过在本地单元注册表中创建影子主体来完成。影子主体是连接有新的扩展注册表属性的主体。这个 ERA 包含一个完全不同的 DCE 单元名称，该名称代表定义外部用户帐户的单元。然后，安全性集成使用主单元 ERA 的值来与外部单元联系，并在登录期间认证用户。影子主体充当将认证重定向到在主单元 ERA 中指出的外部单元的指针。

用户通过指定影子主体的名称来登录，并给出主单元 ERA 所指向的外部单元帐户的口令。然后认证用户，如果成功的话，则接收到外部单元用户的 DCE 凭证。在下例中，用户 **henry** 在单元 **wildwest.austin.ibm.com** 中定义了一个帐户，但是想在单元 **windsor.austin.ibm.com** 中配置的主机上登录。影子主体 **windsor.austin.ibm.com** 单元可能如下定义:

```
Name:           henry
Full Name:      Henry D. Shadow
UNIX ID:        112
UUID:           00000070-e63f-21cf-88-02608c2f5cec
Home Cell ERA:  /.../wildwest.austin.ibm.com
```

影子用户 **henry** 的成功登录将使登录用户具有单元 **wildwest.austin.ibm.com** 中用户 **henry** 的凭证。

主单元 ERA 可能也包含完全不同的 DCE 用户名，包含全部单元名和外部单元用户名。这就允许影子主体用一个不同的名称表示外部单元用户。使用前面的示例，如果主单元 ERA 是 **/.../wildwest.austin.ibm.com/joe**，则 **henry** 的成功登录将获得单元 **wildwest.austin.ibm.com** 中用户 **joe** 的 DCE 凭证。

单元间管理-- 安全性集成

在登录期间，用安全性集成来从 DCE 注册表检索本地 AIX 用户和组信息。因为这个，所以管理员需要理解如何配置和管理 DCE 用户和组，以与单元间环境中的安全性集成一起使用。

影子主体的 UNIX 标识

除了具有外部单元用户的主单元 ERA 之外，影子主体还用来获得登录用户的本地 UNIX 标识。这就允许当本地单元与外部单元中的标识冲突时重新映射用户 ID。影子主体无需具有与主单元 ERA 所指向的外部单元相同的用户 ID。用户登录并接收影子具有的用户标识。

帐户信息

如果在本地单元创建了影子主体的帐户（影子帐户），则使用该帐户来获得本地 AIX 帐户信息。这包括用户的主组、组集、登录外壳、主目录和 gecos 信息。否则，通过主单元 ERA 的方式直接从外部单元检索影子主体的帐户信息。如果本地单元管理员信任外部单元注册表，则无需创建本地影子帐户。

即使影子主体具有影子帐户，当它们使用安全性集成登录时，它们通过主单元 ERA 所指向的外部单元帐户被认证。

影子组

如果从外部单元帐户获得了用户的组集（例如，他们在本地单元中没有影子帐户），则必须在本地单元注册表中创建用户组集中所有组的影子组。影子组与影子主体类似，具有指向实际定义组的外部单元的主单元 ERA。影子组自身没有成员。影子组必须也具有与其主单元 ERA 所指向的外部组相同的组标识。

如果用户具有本地影子帐户，则从本地帐户获得其组集。组必须在本地单元中存在。

管理单元中的 UNIX 标识

如果要在单元间环境中只使用安全性集成，则建议尽可能地分开单元中的用户和组 UNIX 标识。每个单元只在特定于单元的特定范围内分配 DCE 用户和组标识。这就使影子组和用户的管理容易得多。可以在任何外部单元中用与其主单元中相同的标识创建影子，而无需考虑是否已经分配了 ID。

使用以下步骤来设置单元间安全性集成:

1. 建立 DCE 单元之间的信任关系

   要在单元间环境中使用 AIX/DCE 安全性集成，必须在涉及的单元之间建立信任关系。

2. 用单元间标志启动 dceunixd 守护程序

   必须用 **-i 1** 标志启动 dceunixd 守护程序，以使它能够识别主单元 ERA，并与外部单元联系以进行单元间操作。将 **-i 1** 标志添加到为 dceunix 指定的选项。如果守护程序已经在运行，则停止它，然后再重新启动。

3. 定义主单元扩展注册表属性的模式

   在可以将主单元 ERA 的任何实例连接到影子主体或组之前,必须在注册表中定义 ERA 的模式。这需要在单元中、并且只在要创建影子的单元中仅执行一次。以下 dcecp 命令将创建主单元 ERA。必须用与下例所指定的完全一致的值来输入命令。必须具有单元中的 **cell_admin** 特权来执行命令。

   ```
   xattrschema create /.:/sec/xattrschema/shadow_home_cell -attribute {
           {aclmgr
                   {{principal r m m m}
                     {group r m m m}}}
           {annotation {ERA for intercell integrated security} }
           {applydefs no}
           {encoding printstring}
           {intercell accept}
           {multivalued no}
           {reserved no}
           {trigtype none}
           {unique no}
           {uuid 35c6cba-e3d2-11cf-b772-268c2f5cce}
   }
   ```

   当创建 **shadow_home_cell** 模式时，必须使用这个 uuid: "**35c6cba-e3d2-11cf-b772-268c2f5cce**"。

4. 在本地单元中创建影子主体

影子主体与普通的 DCE 主体类似，但是它连接有一个指向外部单元用户的主单元 ERA 实例。一旦创建了影子主体，就可以使用 dcecp 命令将主单元 ERA 连接到它们。

```
dcecp> principal modify name -add "shadow_home_cell /.../cellname"
```

5. 在本地单元中创建影子组

   如果要从影子主体的外部单元帐户获得影子主体帐户信息，则为用户组集中的 gruop 创建影子组。影子组连接有一个指向实际定义组的外部单元的主单元 ERA 实例。可以使用 dcecp 命令将主单元 ERA 连接到它们。

   ```
   dcecp> group modify name -add "shadow_home_cell /.../cellname"
   ```

   影子组必须用与其主单元 ERA 所指向的外部组相同的组标识创建。

6. 在本地单元中创建影子帐户

   除非在本地单元中创建主体的帐户，否则影子主体的帐户信息从其外部单元帐户获得。如果单元管理员信任外部单元注册表中的帐户信息，则无需创建影子帐户。

7. 配置集成安全性操作。

   有关如何配置集成安全性操作系统的详细信息，请参阅第409页的『步骤--安全性集成』。

瘦客户机配置上的安全性集成

为了在瘦客户机上使用安全性集成功能，必须用 **-s** 标志启动 dceunixd 守护程序。如果试图在瘦客户机上不用该标志来启动 dceunixd 守护程序，将得到以下错误信息:

(dceunixd) 确保存在有效的机器凭证，并且没有设置 **KRB5CCNAME** 环境变量。

因为 **dced** 不在瘦客户机配置上运行，所以没有可用的确证服务。用 **-s** 标志启动 dceunixd。

将确认用户的标识，但是将不会证明它们。

passwd_override 和 **group_override** 功能在具有瘦客户机配置的系统上将不可用。

与多个 Dceunixd 守护程序的安全性集成

用 **-n numdaemons** 标志最多可启动五个 dceunixd 守护程序。缺省的（不带 **-n numdaemons** 标志）是一个 dceunixd 守护程序。多个 dceunixd 守护程序增加了那些具有大量用户的系统的安全性集成容量。

在当前发行版中，进程不监控多个 dceunixd 守护程序。这意味着，如果出于某种原因要杀死 dceunixd，则管理员将需要手工杀死所有以前可能用 dceunixd **-n numdaemons** 标志启动的 dceunixd 守护程序。

-n numdaemons 和 **-d level** 标志是互斥的。

第42章 可插入认证模块（仅 Solaris）

“可插入认证模块”(PAM) 是将多认证机制集成到标准系统程序的 API 和框架。PAM 模块是位于 **/usr/lib/security** 目录中的共享资源库。PAM 库文件在安装和配置过程期间自动安装到 **/usr/lib** 目录中。这些共享资源库包含三种模块类型--**authentication**、**account** 和 **password**。

注: 由于操作系统限制，PAM 在 Solaris 2.5.1 或更早版本上不可用。如果正在 Solaris 2.5.1 或更早版本上运行 DCE * DFS，请使用 **login.dce** 命令。

sso_cell 配置文件

PAM 使用 **/opt/dcelocal/etc/security** 中的 **sso_cell** 配置文件来确定用户归属的单元（如果没有命名其它单元）。该文件还用来确定用户归属于缺省单元（用 **defaultcell** 标志列出的）、本地单元（在其中配置机器的单元）或一个替代单元（用 **altcells** 标志列出）。

注: **sso_cell** 文件必须由 **root** (UID **0**) 所有并且设置 **0644** 方式位 (**-rw-r--r--**)。如果文件与这两项需求不符，将忽略文件并记录警告。

使用 **config.dce pam** 命令在 **sso_cell** 文件中定义缺省单元。**defaultcell** 标志将设置如下:

```
defaultcell = /.../your.cell.name
```

unconfig.dce pam 将除去 **sso_cell** 文件。

要定义一个或多个使用 **sso_cell** 文件的替代单元，请将 **altcells** 标志设置如下:

```
altcells = /.../alt.cell.name,
/.../another.cell.name
```

编辑 **sso_cell** 配置文件时，允许空白行、空白和注释。

Authentication 模块

authentication 模块验证用户口令，建立用户凭证文件和 “DFS 进程认证组”(PAG)，并可任选地生成口令失效警告。它接受下列 **pam.conf** 文件配置选项:

- **use_first_pass** 选项使认证尝试只能够使用为以前的模块提供的口令。只有当首先调用认证模块时，使用该选项才启用口令提示。
- **try_first_pass** 选项允许认证尝试使用为该系列中以前的模块提供的口令。如果尝试失败或不存在以前的模块，将出现需要 DCE 口令的提示。
- **ignore_root** 选项禁用名为 root 或 UNIX UID 为 0 的用户的任何认证尝试。
- **nowarn** 选项禁止生成警告，包括口令失效警告。
- **warnexpire=***N* 选项（其中 *N* 是正整数），如果用户口令在 *N* 天之内失效，它将发出警告。缺省设置是 30 天。
- **bind_pe_site option** 使 PAM 能够只使用 **/opt/dcelocal/etc/security/pe_site** 文件来定位注册表服务器。该选项的设置类似于设置 **BIND_PE_SITE** 环境变量。

- **proxiable** 标志允许使用带有不同网络地址的票券而不是必须使用从 **auth** 模块发到用户的现有票券。为授予 proxiable 票券，注册表中的用户帐户必须指定能接受 proxiable 票券的用户。
- **forwardable** 标志允许使用带有不同网络地址的授予票券的票券 (TGT)，而不是必须使用要发到主体的现有 TGT。为了授予转发的票券，注册表中的用户帐户必须指定能接受转发票券的主体。

authentication 模块的推荐配置是 **try_first_pass ignore_root**。

Account 模块

account 模块可任选地检查 **getpwnam** 和 DCE 注册表以确保二者都承认 UNIX UID，检查用户 login context 证书并确认用户对于 DCE hostdata 对象的正常许可权。**login** 程序的推荐配置是 **ignore_root require_match require_cert nondce_ok**。

注: 指派 **account** 模块时忽略 **asswd_override** 文件。

- **require_match** 选项禁止登录访问任何这样的用户，它们的 UNIX UID 是通过系统 **getpwnam_r** 函数（它检查 **/etc/passwd** 文件以获取信息）获得的并且与存储在 DCE 注册表中的 UNIX UID 不匹配。
- 如果 DCE login context 不到授权，**require_cert** 选项禁止访问 DCE。运行认证模块运行后如果有效的 UID 为 0，则将授权通过 DCE PAM 获得的 login context。为系统 login 进程指定 **require_cert**，例如 **login**、**telnet**、**dtlogin** 或本地 **root** 执行的任何 PAM 应用程序。
- **ignore_root** 选项禁用名为 **root** 或其 UNIX UID 为 0 的用户的任何认证尝试。
- **nowarn** 选项禁止生成警告，包括口令失效警告。
- **nondce_ok** 选项允许那些没有 DCE login context 和在 DCE 注册表中不存在的用户访问 DCE。如果用户在 DCE 注册表中存在但没有 login context，将拒绝访问。
- **check_acl** 选项确认带有 UUID 14d5e798-2f75-11d1-9c40-9e62070caa7 的本地主机读访问 DCE hostdata 对象 ACL 并授权 DCE 访问。
- **alt_acl=***UUID* 选项，其中 *UUID* 是本地 hostdata 对象的 UUID，将检查任何备用的 DCE hostdata 对象上的 ACL。使用该选项作为替代项或与 **check_acl** 选项结合以检查 ACL 或者一次分别进行的双重 ACL 检查。用相同步骤为 **login**、**ftp**、通过 **dtlogin** 的控制台访问和 **telnet** 等设置不同的 ACL。还可以通过设置 **alt_acl** 选项替换 **check_acl** 选项来检查带有任何其它 UUID 的相似对象。
- **bind_pe_site** 选项允许 PAM 定位仅使用 **/opt/dcelocal/etc/security/pe_site** 文件的注册表服务器。该选项的设置类似与设置 **BIND_PE_SITE** 环境变量。

要检查使用 PAM 登录的 ACL。

1. 使用下列命令启用 **dced hostdata** 接口:

   ```
   $dced -x
   ```

   注: 必须使用 **-x** 选项使 **dced hostdata** 可用。如果 **dced hostdata** 不可用或 hostdata 对象不存在，将拒绝访问。
2. 使用下列在带有特定 UUID 的主机上创建 **hostdata** 对象。

```
dcecp -c hostdata create /.:/hosts/machine/config/hostdata/login_svc\
-storage /opt/dcelocal/var/security/login_svc -data { {} }
-uuid 14d5e798-2f75-11d1-9c40-9e62070caa7
-annotation {The ACL on this object determines login permissions for this
host.}
```

3. 将 **check_acl** 变量添加到 **pam.conf** 文件的 **pam_dce** 帐户行中。

   注: r 许可权确定登录许可权。

Password 模块

password 模块更改 DCE 注册表中的用户口令，宽松地执行 DCE 口令强度策略，并在更改口令时任选地刷新用户的 DCE 凭证。

根据服务在 **/etc/pam.conf** 文件中列出了每个模块类型。服务的示例是 **login**、**ftp** 和 **dtlogin**。因为按出现在 **pam.conf** 文件中的顺序读取模块，可以堆积或列出每种类型和服务的多个模块。还可以将模块指定为必需的，可选的或足够的。

要使用 PAM，必须通过插入指定 DCE 资源库的行来编辑 PAM 配置文件 **/etc/pam.conf**。将这些行紧接着放置在指定 UNIX 资源库的类似行后。**pam.conf** 文件中的一个示例是:

```
# Authentication management#
login   auth optional   /usr/lib/security/pam_dce.so.1 try_first_pass ignore_root
rlogin  auth optional   /usr/lib/security/pam_dce.so.1 try_first_pass ignore_root
dtlogin auth optional   /usr/lib/security/pam_dce.so.1 try_first_pass ignore_root
other   auth optional   /usr/lib/security/pam_dce.so.1 try_first_pass ignore_root
# Account management
login   account optional /usr/lib/security/pam_dce.so.1 ignore_root require_match
dtlogin account optional /usr/lib/security/pam_dce.so.1 ignore_root require_match
other   account optional /usr/lib/security/pam_dce.so.1 ignore_root require_match
# Password management
other   password optional /usr/lib/security/pam_dce.so.1 refresh
```

DCE 应用程序调用 PAM API 并链接到 PAM 资源库 **libpam.so.1** 时， PAM 框架找到适当的模块，装入该模块并调用由模块提供的 PAM 服务器提供者接口 (SPI) 函数。

如果将 “可插入认证模块” 配置成使用 DCE 提供的资源库，则必须在迁移进程期间重新启动系统前修改 **/etc/pam.conf** 文件来除去 DCE 资源库的引用。**dcesetup upgrade_uninstall** 命令除去 **/etc/pam/.conf** 中 DCE 资源库的引用。如果未作修改，将无法在重新启动后登录系统，因为 PAM 试图找到在升级卸载期间除去的 DCE 资源库。

如果在迁移后使用 PAM，可在除去 DCE 资源库的引用前备份 **/etc/pam.conf**。然后，在完成 **dcesetup upgrade_install** 进程后恢复它。

以下是缺省的 **/etc/pam.conf** 文件示例，该文件是在只引用 UNIX 资源库的 Solaris 7 上配置的。

```
#
# PAM configuration
#
# Authentication management
#
login   auth required   /usr/lib/security/pam_unix.so.1
login   auth required   /usr/lib/security/pam_dial_auth.so.1
#
rlogin  auth sufficient /usr/lib/security/pam_rhosts_auth.so.1
```

```
rlogin  auth required   /usr/lib/security/pam_unix.so.1
#
dtlogin auth required   /usr/lib/security/pam_unix.so.1
#
rsh     auth required   /usr/lib/security/pam_rhosts_auth.so.1
other   auth required   /usr/lib/security/pam_unix.so.1
#
# Account management
#
login   account required      /usr/lib/security/pam_unix.so.1
dtlogin account required      /usr/lib/security/pam_unix.so.1
#
other   account required      /usr/lib/security/pam_unix.so.1
#
# Session management
#
Other   session required      /usr/lib/security/pam_unix.so.1
#
# Password management
#
Other   password required    /usr/lib/security/pam_unix.so.1
```

如果忘了除去 **/etc/pam.conf** 中 DCE 资源库的引用和重新启动系统，则无法登录系统。在这种情况下，可以以单一用户方式重新启动系统。保存 **/etc/pam.conf** 的备份。从 **etc/pam.conf** 除去 DCE 资源库的引用。以正常方式重新启动。登录以完成剩余迁移。然后，恢复 **/etc/pam.conf** 的备份再次启用 PAM 功能。

第43章 访问注册表对象

本章描述应用于注册表中对象的许可权。因为授予的许可权基于注册表数据库的结构方式，所以本章先简要地描述注册表数据库的结构。然后，再描述注册表数据库、注册表 ACL 管理器和初始注册表 ACL 中每个对象的许可权。

dcecp 和 **acl_edit** 命令都有创建、修改和删除注册表对象的 ACL 项功能。请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference* 中的每个命令，以获得它对 ACL 项执行的操作描述。

注册表数据库

注: 关于 “DCE 安全性注册表” 和 “LDAP 集成” 特性的主题信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: DCE Security Registry and LDAP Integration Guide*。

注册表结构分成下列主目录:

- **principal** 目录--包含有关主体的信息
- **group** 目录--包含有关组的信息
- **org** 目录--包含有关组织的信息

除了目录外，注册表还包含 **policy** 对象、**replist** 对象和 **xattrschema** 对象，它们都是在机器配置期间创建注册表时创建的。**policy** 对象包含应用于注册表特性以及策略和组织策略的信息；**replist** 对象包含有关 DCE 单元中的副本信息；**xattrschema** 对象包含有关扩展注册表属性 (ERA) 的信息。通过使用 **dcecp registry** 命令，可以在任何时候修改**策略**和**副本**信息。使用 **dcecp xattrschema** 命令，可以修改 **xattrschema** 对象。

当在 **principal**、**group** 或 **org** 目录中创建简单对象时，根据需要创建子目录。例如，如果添加主体（如 **preludes/villa/lobos**），则创建子目录 **preludes** 和 **villa**。可以使用这些子目录来帮助组织数据。当删除子目录中的所有对象时，子目录本身也被删除。（不能删除 **principal**、**group** 或 **org** 目录。）

授予注册表中对象的许可权取决于该对象在注册表数据库结构中所处的位置。图54 说明了注册表数据库。矩形框表示容器对象（目录）。椭圆形框表示简单对象。图54 只显示顶层 **principal**、**group** 和 **org** 目录。如果您创建子目录的话，注册表可以有子目录。

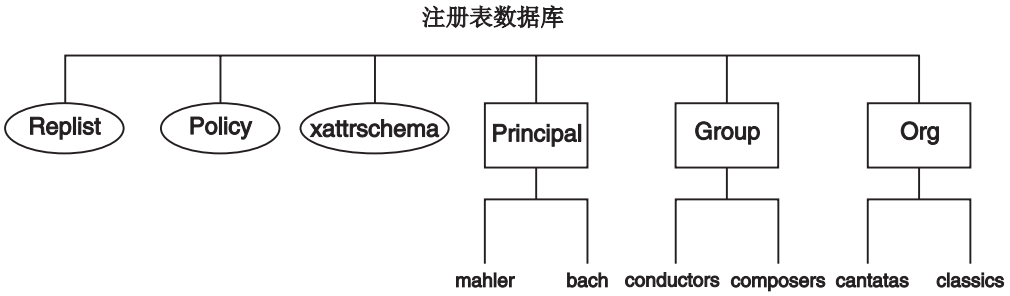


图 54. 注册表数据库结构

注册表许可权

表30 列出了可授予在注册表中找到的对象类型的许可权。

表 30. 注册表对象的许可权

许可权	含义
A	执行作用于副本的命令（**sec_admin**）。
a	修改认证信息。
c	修改对象的 ACL。所有注册表 ACL 都必须有一个指定 **c**（控制）许可权的项。
d	从对象的内容中删除。
D	从注册表删除对象。
f	修改主体、组或组织的全名。
g	将主体添加到组。
i	添加到对象的内容。
m	修改管理信息。
M	将成员添加到这个组或组织，以及从这个组或组织删除成员。要将成员添加到组，对于要添加的主体还必须有 **g** 许可权。
n	修改目录、主体、组或组织的名称。
u	修改用户信息。
r	查看管理、认证和用户信息。
t	测试已命名主体的组或组织成员。

管理、认证和用户信息

注册表包含三种不同有关其中对象的信息: 管理信息、认证信息和用户信息。在下面部分中，将概述用于每个对象类型的信息的特定项。

管理信息

管理信息包括下列类别:

- 对于注册表策略和特性:
  - 帐户使用期限
  - 口令的最小长度
  - 口令使用期限
  - 口令是否可以包含空格
  - 口令是否可以完全由非字母数字字符组成
  - 口令失效日期
  - 最小的票券寿命
  - 缺省票券寿命
  - 定义最小 UNIX 标识的数字，该标识是在创建主体、组或组织时自动提供的。
  - 定义最大号的数字，在创建主体。组或组织时可自动或手工提供它，以作为 UNIX 标识。
  - 是否显示加密口令（阴影口令特性）
- 对于主体:
  - 帐户、组和组织名称
  - 显示主体全名的文本字符串

  - 用于主体的对象创建限额
  - 主体是否可以将主名称更改为别名并且将别名更改为主名称
  - 主体的用户标识 (UID)
  - 主体的唯一用户标识 (UUID)
  - 主体帐户的失效日期
  - 主体帐户的帐户有效标志
  - 标志，它指示帐户用于作为客户机的主体还是用于作为服务器的主体
- 对于组:
  - 组的主名称
  - 显示组全名的文本字符串
  - 是否可以将组的主名称更改为别名并且将它的别名更改为它的主名称
  - 组的组标识 (GID)
  - 项目列表包含特性
  - 组的 UUID
- 对于组织:
  - 组织的主名称
  - 是否可以将组织的主名称更改为别名并且将它的别名更改为它的主名称
  - 显示组织全名的文本字符串
  - 组织的组织标识 (ORGID)
  - 组织的 UUID
  - 帐户使用期限
  - 口令的最小长度
  - 口令使用期限
  - 口令失效日期
  - 口令是否可以包含空格
  - 口令是否可以完全由非字母数字字符组成
- 对于 **xattrschema** 对象:
  - 是否可以修改 xattrschema

认证信息

认证信息包括下列类别:

- 对于注册表策略和特性:
  - 最大的票券寿命
  - 用于更新票券的最大时间
- 对于主体:
  - 主体帐户的最大票券寿命
  - 用于更新向主体帐户发出的票券的最大时间
  - 最后一次更改主体帐户的日期和时间（最佳开始日期）
  - 启用主体帐户的日期和时间（最后一次更改日期）
  - 主体帐户的创建者和帐户创建日期
  - 帐户使用的描述
  - 是否可以向主体帐户发出推迟日期票券、可转寄票券、可更新票券或可代理票券。
  - “DCE 认证服务”是否可以根据票券授予票券认证向主体帐户发出票券，或者主体是否必须获得服务的直接票券
  - 是否可以向主体帐户发出重复会话密钥

用户信息

用户信息包括下列有关主体帐户的信息:

- 口令
- 主目录
- 多种信息（GECOS 信息）
- 注册外壳
- 口令有效标志

创建主体、组或组织所需的许可权

图55 显示创建主体、组或组织所需的许可权。

图 55. 创建主体、组或组织所需的许可权

要创建主体、组或组织，对要在其中创建主体、组或组织的目录，必须有 **i** 许可权。例如，要创建主体 **preludes/villa/lobos**，对 **villa** 必须有 **i**。

删除主体、组或组织所需的许可权

图56 显示删除主体、组或组织所需的许可权。

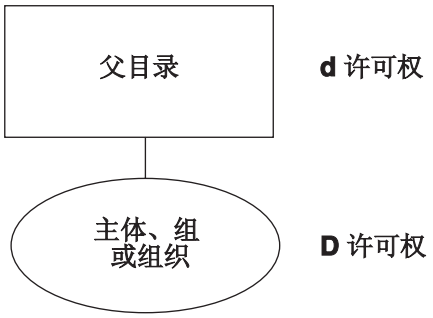


图 56. 删除主体、组或组织所需的许可权

要删除主体、组或组织，必须有下列许可权:

- 对要删除的主体所在目录有 **d** 许可权
- 对要删除的主体、组或组织有 **rD** 许可权

例如，要删除主体 **preludes/villa/lobos**，必须对 **preludes/villa** 目录有 **d** 许可权，对主体 **preludes/villa/lobos** 有 **rD**许可权。

添加帐户所需的许可权

当使用 **user create** 命令添加帐户时，如果主体还不是组和／或组织的成员，则 **dcecp** 将主体添加到在帐户中命名的组或组织。由于这个原因，添加帐户所需的许可权可能包括将成员添加到组或组织所需的许可权。下列主题涉及添加帐户所需的许可权讨论:

- 添加帐户同时将主体作为在帐户中命名的组和组织的成员添加所需的许可权。（请参阅第422页的『添加帐户所需的许可权』。）
- 为已是命名组和组织的成员的主体添加帐户所需的许可权。（请参阅『为已是组和组织成员的主体添加帐户』。）
- 添加帐户并且只将主体添加到在帐户中命名的组（因为主体已是组织的成员）所需的许可权。（请参阅第424页的『只将帐户和主体添加到组』。）
- 添加帐户并且只将主体添加到在帐户中命名的组织（因为主体已是组的成员）所需的许可权。（请参阅第425页的『只将帐户和主体添加到组织』。）

将帐户和帐户主体添加到组和组织

图57 显示了将帐户和帐户主体添加到组或组织所需的许可权。

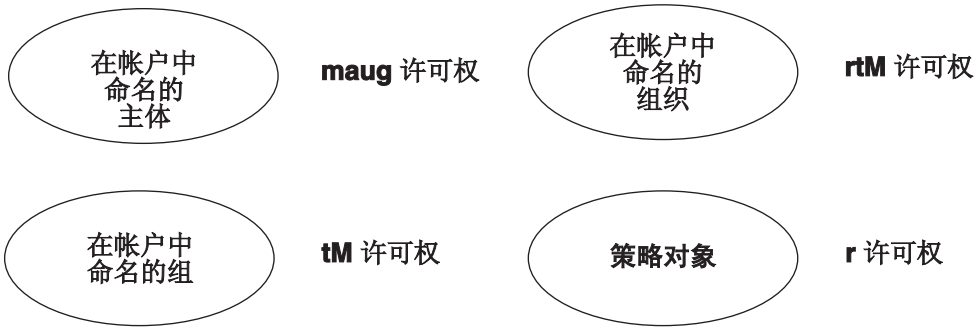


图 57. 将帐户和帐户主体添加到组和组织所需的许可权

要添加一个帐户并将帐户的主体添加到在帐户中自动命名的组和组织，必须有下列许可权:

- 对帐户主体有 **maug** 许可权
- 对在帐户中命名的组有 **tM** 许可权
- 对在帐户中命名的组织有 **rtM** 许可权
- 对**策略**对象有 **r** 许可权

例如要为与组 **composers** 和组织 **pianists** 相关的主体 **preludes/villa/lobos** 创建帐户，必须有下列许可权:

- 对 **preludes/villa/lobos** 有 **maug** 许可权
- 对组 **composers** 有 **tM** 许可权
- 对组织 **pianists** 有 **rtM** 许可权
- 对**策略**对象有 **r** 许可权

为已是组和组织成员的主体添加帐户

第424页的图58 显示了为已是组或组织成员的主体添加帐户所需的许可权。

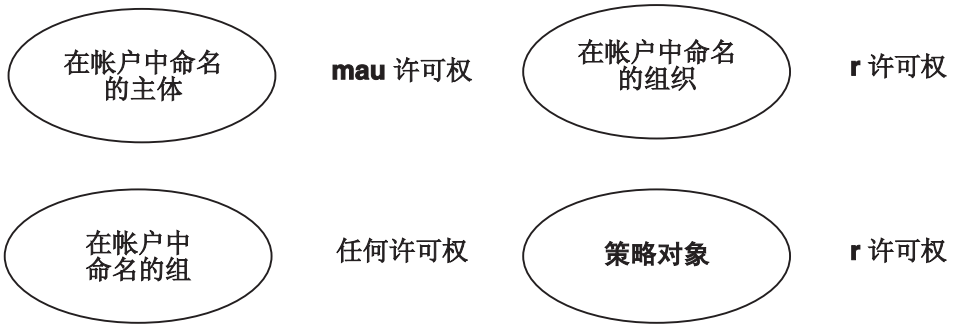


图 58. 为已是组和组织成员的主体添加帐户

要添加一个帐户并且不需要将帐户主体添加到在帐户中命名的组和组织，必须有下列许可权:

- 对帐户主体有 **mau** 许可权
- 对在帐户中命名的组至少有一种任何类型的许可权
- 对在帐户中命名的组织有 **r** 许可权
- 对**策略**对象有 **r** 许可权

例如要为与组 **composers** 和组织 **pianists**相关的主体 **preludes/villa/lobos** 创建帐户，必须有下列许可权:

- 对 **preludes/villa/lobos** 有 **mau** 许可权
- 对组 **composers** 至少有一种任何类型的许可权
- 对组织 **pianists** 有 **r** 许可权
- 对**策略**对象有 **r** 许可权

只将帐户和主体添加到组

图59 显示了只将帐户和主体添加到组所需的许可权。

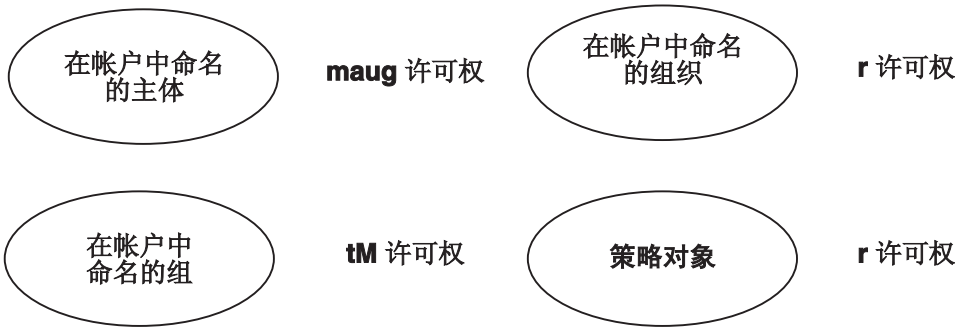


图 59. 只将帐户和主体添加到组所需的许可权

要添加一个帐户并将帐户的主体添加到组（主体已是在帐户中命名的组织的成员），必须有下列许可权:

- 对帐户主体有 **maug** 许可权
- 对在帐户中命名的组有 **tM** 许可权
- 对在帐户中命名的组织有 **r** 许可权
- 对**策略**对象有 **r** 许可权

只将帐户和主体添加到组织

图60 显示了只将帐户和主体添加到组织所需的许可权。

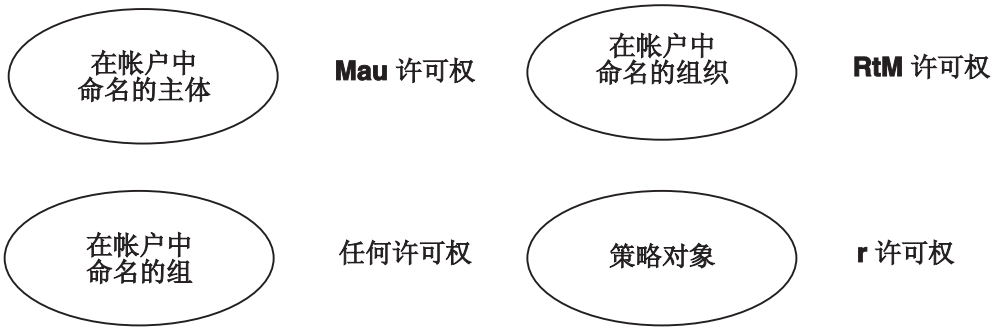


图 60. 只将帐户和主体添加到组织所需的许可权

要添加一个帐户并将帐户的主体添加到组织（主体已是在帐户中命名的组的成员），必须有下列许可权:

- 对帐户的主体有 **mau** 许可权
- 对在帐户中命名的组至少有一种任何类型的许可权
- 对在帐户中命名的组织有 **rtM** 许可权
- 对**策略**对象有 **r** 许可权

删除帐户所需的许可权

图61 显示了删除帐户所需的许可权。

图 61. 删除帐户所需的许可权

要删除帐户，对在帐户中命名的主体必须有 **rmau** 许可权。例如，要增删名为 **preludes/villa/lobos** 的主体的帐户，对 **preludes/villa/lobos** 必须有 **rmau** 许可权。

将成员添加到组所需的许可权

图62 显示了将成员添加到组所需的许可权。

图 62. 将成员添加到组所需的许可权

要将成员添加到组，必须有下列许可权:

- 对正在添加主体的组有 **rM** 许可权
- 对要添加的主体有 **rg** 许可权

例如，要将主体 **preludes/villa/lobos** 添加到组 **composers**，必须有下列许可权:
- 对组 **composers** 有 **rM** 许可权
- 对主体 **lobos** 有 **rg** 许可权

将成员添加到组织所需的许可权

图63 显示了将成员添加到组织所需的许可权。

图 63. 将成员添加到组织所需的许可权

要将成员添加到组织，必须有下列许可权:
- 对正在添加主体的组织有 **rM** 许可权
- 对要添加的主体有 **r** 许可权

例如，要将主体 **preludes/villa/lobos** 添加到组织 **pianists**，必须有下列许可权:
- 对组织 **pianists** 有 **rM** 许可权
- 对主体 **lobos** 有 **r** 许可权

从组或组织删除成员所需的许可权

图64 显示了从组或组织删除成员所需的许可权。

图 64. 从组或组织删除成员所需的许可权

要从组或组织删除成员，对要从其中删除主体的组或组织必须有 **rM** 许可权，对正在删除的主体必须有 **r** 许可权。

例如，要从组 **composers** 删除主体 **preludes/villa/lobos**，必须有下列许可权:
- 对组 **composers** 有 **rM** 许可权
- 对主体 **lobos** 有 **r** 许可权

更改主体、组或组织的全名所需的许可权

第427页的图65 显示了更改主体、组或组织的全名所需的许可权。

图 65. 更改主体、组或组织的全名所需的许可权

要更改主体、组或组织的全名，对要进行更改的主体、组或组织必须有 **rf** 许可权。

更改主体、组或组织的管理信息所需的许可权

图66 显示了更改主体、组或组织的管理信息所需的许可权。

图 66. 更改主体、组或组织的管理信息所需的许可权

要更改主体、组或组织的管理信息，对正在更改其管理信息的对象必须有 **rm** 许可权。

更改帐户的管理、认证和用户信息（除了口令）所需的许可权

图67 显示了更改帐户的管理、认证和用户信息（除了口令）所需的许可权。

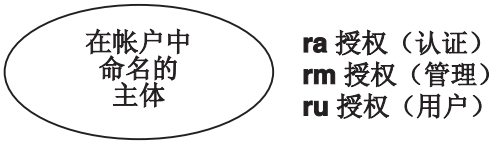


图 67. 更改帐户的管理、认证和用户信息（除了口令）所需的许可权

要更改帐户的所有管理、认证和用户信息（除了口令），对在帐户中命名的主体必须有下列许可权:

- 更改认证信息的 **ra** 许可权
- 更改管理信息的 **rm** 许可权
- 更改用户信息的 **ru** 许可权

更改帐户口令所需的许可权

图68 显示了更改帐户口令所需的许可权。

图 68. 更改帐户口令所需的许可权

要更改帐户口令，对在帐户中命名的主体必须有下列许可权:

- 对帐户的主体有 **ru** 许可权

- 对**策略**对象有 **r** 许可权

更改注册表策略和特性的认证和管理信息所需的许可权

图69 显示了更改注册表策略和特性的认证和管理信息所需的许可权。

图 69. 更改注册表策略和特性的认证和管理信息所需的许可权

要通过使用 **dcecp registry modify** 命令更改管理或认证信息，对注册表**策略**对象必须有更改认证信息的 **ra** 许可权或更改管理信息的 **rm** 许可权。

执行作用于副本的命令所需的许可权

图70 显示了执行作用于副本的命令所需的许可权。

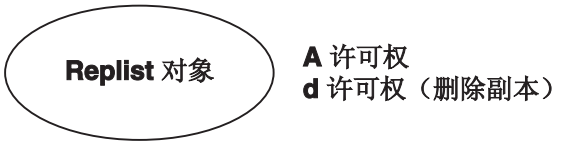


图 70. 执行作用于副本的命令所需的许可权

要执行任何作用于副本的命令，对 **replist** 对象必须有下列许可权:

- 执行所有命令的 **A** 许可权，除了显示副本信息的那些命令，它们对 **replist** 对象不需要许可权。
- 执行删除副本的命令的 **d** 许可权。

创建扩展注册表属性类型所需的许可权

图71显示创建 ERA 类型所需的许可权。

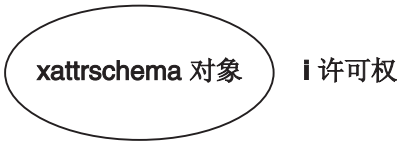


图 71. 创建扩展注册表属性类型所需的许可权

要在注册表模式中创建 ERA 类型，对 **xattrschema** 对象必须有 **i** 许可权。

删除扩展注册表属性类型所需的许可权

第429页的图72显示删除 ERA 类型所需的许可权。

d 许可权

图 72. 删除扩展注册表属性类型所需的许可权

要删除 ERA 类型，对 **xattrschema** 对象必须有 **d** 许可权。

查看扩展注册表属性类型所需的许可权

图73 显示了查看注册表的模式数据库中的一个或多个 ERA 所需的许可权（使用 **dcecp xattrschema show** 命令）。

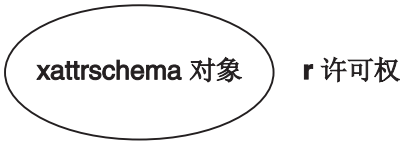


图 73. 查看扩展注册表属性所需的许可权

要查看 ERA 类型，对 **xattrschema** 对象必须有 **r** 许可权。

修改扩展注册表属性类型所需的许可权

图74 显示了修改 ERA 类型所需的许可权。

图 74. 修改扩展注册表属性类型所需的许可权

要修改 ERA 类型，对 **xattrschema** 对象必须有 **m** 许可权。

更改注册表对象的 ACL 所需的许可权

图75 显示了更改注册表对象的 ACL 所需的许可权。

图 75. 更改注册表对象的 ACL 所需的许可权

要修改注册表对象的 ACL，对正在更改其 ACL 的对象必须有 **c** 许可权。注册表对象可以是**策略**对象或主体、组或组织。

从属副本所需的许可权

为了正确地初始化和运行，从属副本必须有 **replist** 对象（**/.:/sec/replist**）的 **i**、**m** 和 **l** 许可权。从属服务器在运行它的机器的标识下运行。机器名是本地主机主体名，其形式如下:

host/*hostname*/self

当 **dce_config** 工具最初配置 DCE 单元的安全性服务器时以及当使用该工具来创建新从属副本时，添加必需的 ACL 项。项的形式如下:

user:host/*hostname*/self:imI

注册表 ACL 管理器

注册表 ACL 管理器由五种管理器类型组成，它们用于处理注册表中五种对象所需的不同 ACL 语义。例如，主体 ACL 管理器类型控制注册表中所有 **principal** 对象的 ACL。因为 **group** 对象需要一组与 **principal** 对象的许可权不同的许可权，所以有一个单独的组 ACL 管理器类型，用于控制 **group** 对象的 ACL。

不是所有的许可权也不是所有的 ACL 项类型都对每个 ACL 管理器有效。 表31 概述了每个 ACL 管理器的有效和无效许可权以及无效 ACL 项类型。

表 31. ACL 管理器和有效的许可权和 ACL 项类型

管理器类型	控件	有效的许可权	无效的 ACL 项类型
dir	**directory** 对象	**rcidDn**	**user_obj**, **group_obj**
policy	**policy** 对象	**rcma**	**user_obj**, **group_obj**
principal	**principal** 对象	**rcDnfmaug**	**group_obj**
group	**group** 对象	**rctDnfmM**	**user_obj**
org	**org** 对象	**rctDnfmM**	**user_obj**, **group_obj**
replist	副本列表	**cidmlA**	**user_obj**, **group_obj**
xattrschema	ERA 类型	**rcidm**	**user_obj**, **group_obj**

初始注册表 ACL

当创建注册表数据库时，给予 **principal**、**group** 和 **org** 目录和 **policy**、**replist** 和 **xattrschema** 对象初始 ACL。当在注册表中创建新对象时，它们从 **principal**、**group** 和 **org** 目录 ACL 继承它们的 ACL。需要键的初始 ACL 项的 ACL 项键是创建注册表数据库的主体的名称（提供给 **sec_create_db** 命令作为注册表创建者），或如果没有提供名称，则是 **root**。（有关 **sec_create_db** 和注册表创建者的详细信息，请参阅第375页的『 第38章 设置注册表 』。）

下面的列表描述当创建注册表数据库时创建的初始 ACL。在这个列表中，*rgy_creator* 表示命名为注册表创建者的主体。

注: 您的平台配置工具可能更新这些初始 ACL。

- 对于 **principal** 对象:

```
unauthenticated:r--------
user_obj:r---f--ug
user:rgy_creator:rcDnFmaug
other_obj:r-------g
any_other:r--------
```

- 对于 **group** 对象:

```
unauthenticated:r-t-----
user:rgy_creator:rctDnfmM
group_obj:r-t-----
other_obj:r-t-----
any_other:r-t-----
```

- 对于 **org** 对象:

```
unauthenticated:r-t-----
user:rgy_creator:rctDnfmM
other_obj:r-t-----
any_other:r-t-----
```

- 对于 **policy** 对象:

```
unauthenticated:r----
user:rgy_creator:rcma
other_obj:r----
any_other:r----
```

- 对于 **directory** 对象:

```
unauthenticated:r-----
user:rgy_creator:rcidDn
other_obj:r-----
any_other:r-----
```

- 对于 **replist** 对象:

```
user:cell_admin:cidmA-
```

- 对于 **xattrschema** 对象:

```
unauthenticated:r-----
user:cell_admin:rcidm
other_obj:r-----
any_other:r-----
```

第44章 DCE 审计服务

审计在分布式系统中起重要作用。检测和记录分布式应用程序中的重要事件需要适当的审计设施。

审计是 DCE 的关键组件，由“DCE 审计服务”提供。本章提供“DCE 审计服务”简介。

DCE 审计服务的功能

“DCE 审计服务”有下列功能:

- 审计守护程序 (**auditd**) 根据指定的标准来记录审计记录。
- 应用程序编程接口 (API) 可以作为应用程序服务器程序来记录审计事件。这些 API 还可用于创建可以分析审计记录和踪迹的工具。
- 至审计守护程序的管理命令接口命令守护程序根据某些标准选择要记录的事件。这个接口是通过 DCE 控制程序 (**dcecp**) 或审计控制 API 访问的。
- 为了便于管理，事件分类机制允许对一组事件进行合理分组。
- 审计记录可以指向日志或控制台。

DCE 审计服务的组件

“DCE 审计服务”有这些基本组件:

- 应用程序编程接口 (API)

  当应用程序服务器服务于客户机时，提供用于检测和记录重要事件的功能。应用程序员对应用程序服务器程序中的某些*代码点*使用这些功能，以启动审计事件的记录。其它 API 可用于创建检查和分析审计事件记录的工具。
- 审计守护程序

  审计守护程序提供下列服务:
  - 维护过滤器和中央审计跟踪文件。
  - 导出 RPC 接口，DCE 控制程序 (**dcecp**) 可以通过这个接口控制它。

DCE 审计服务概念

本部分描述与“DCE 审计服务”的管理相关的一些概念。

审计客户机

所有基于 RPC 的服务器都可能是审计客户机；DCE 服务器和用户编写的应用程序服务器。 “DCE 安全性服务”和“分布式时间服务”是可以审计的。即，代码点（在下一部分中讨论）已处于这些服务中的适当位置。

注: 因为审计子系统的集中式性质，所以建议不要从不起 RPC 服务器作用的 RPC 客户机生成审计事件。因为在典型的客户机服务器应用程序中，服务器的数量可能远远少于客户机的数量，所以对服务器方的审计会限制审计记录将驻留的位置数并易于从审计跟踪收集信息。

然而，只要客户机不尝试使用过滤器（在环境中使用 **DCEAUDITFILTERON** 设置启动），就会对客户机方进行审计。不是 RPC 服务器的 RPC 客户机不能使用过滤，因为它们无法知道过滤器状态的更改时间且不能适合地反映更改。这种能力对于审计子系统的功能需求十分重要。

审计守护程序也可以审计它本身。

审计客户机应该具有对审计守护程序对象的**记录**许可权，以便能够使用中央审计跟踪文件。对审计守护程序的许可权将在第443页的『第45章 DCE 审计服务管理任务』中讨论。

代码点

代码点是应用程序服务器程序中使用 DCE 审计 API 的位置。通常，代码点符合由需要审计的应用程序服务器提供的操作或功能。例如，如果银行服务器提供现金提取功能 **acct_withdraw( )**，则这个功能可能被认为是一个可查找事件并指定为一个代码点。

代码点已处于“DCE 安全性服务”、“分布式时间服务”和“审计服务”代码中的适当位置。 “DCE 安全性服务”的代码点及其相关事件在 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference* 中有描述“DCE 分布式时间服务”的代码点及其相关事件在 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference* 中有描述。 “DCE 审计服务”的代码点及其相关事件在*IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference*中有描述。

审计事件

审计事件是审计客户机希望记录的任何事件。通常，审计事件涉及系统的完整性。例如，当客户从他的银行帐户提取现金时，这是一个审计事件，因为它可能涉及银行帐户的安全性违规。

审计事件与应用程序服务器代码中的代码点相关。

事件编号

应用程序员为每个审计事件指定一个事件编号。事件编号是一个 32 位整数，如 0xC0000000。在 *IBM DCE Version 3.2 for AIX and Solaris: Application Development Guide--Core Components* 中详细描述了事件编号。

事件类

审计事件可以被合理地分组到一个事件类中。事件类提供一种有效机制，由单一值指定一组事件。通常，事件类由具有某些共同性质的审计事件组成。例如，在银行服务器程序中，现金事务（存款、提取和转帐）可以分组为一个事件类。事件类将在第443页的『第45章 DCE 审计服务管理任务』中讨论。

事件类文件

事件类在*事件类文件*中定义。所有事件类文件都必须在 *dcelocal*/**etc/audit/ec** 目录中创建。

提供的缺省事件类文件用于分类“DCE 安全性服务”、“时间服务”和“审计服务”的可查找事件。当安装任何一种服务时，会在主机系统上安装它们。

事件类的名称与它的文件名相同。每个事件类都是在事件类文件内部定义的。

通过除去或添加事件类文件中的事件编号或创建新的事件类文件，可定义新的事件类。

事件类名称

每个事件类都有一个指定的符号名。建议供应商遵循下面的事件类名称格式:

ec_*org_product_class*

其中:

org 是定义事件类的组织或公司的名称。

product
是为其定义事件类的产品的名称。

class 是事件类的特性。

以下是两个事件类名称例子:

- **ec_osf_dce_authentication**--为 OSF 的 DCE 核心组件定义认证事件类。
- **ec_transarc_encina_update**--为 Transarc 的 Encina® 定义更新事件类。

还可定义事件类来满足您自己的审计需要。对于这些事件类，建议使用以下名称格式:

dce_*server-name_class*

其中，*class* 是事件类的特性。

事件类编号

如果定义您自己的事件类，则必须将它们与一个事件类编号相关。事件类编号是 32 位的整数并且对于主机上定义的每个事件类是唯一的。每个事件类编号是由*集合标识*和*类事件标识*组成的元组。集合标识对应于一个事件类集合，并由 OSF 将它指定给组织或供应商。类事件标识标识事件类集合中的一个事件类。组织或供应商管理类事件标识的发布，以生成一个事件类编号。

可以把事件类编号的结构和管理比作 IP 地址的结构和管理。还记得吗: IP 地址是由网络标识（类似于集合标识）和主机标识（类似于类事件标识）组成的元组。

事件类编号格式

根据组织中的事件类数目，事件类编号遵循五个格式（A 到 E）之一。事件类编号的格式可以根据它的四个高位确定。

格式 A 可由大组织（如 OSF 或主要的 DCE 供应商）使用，它们需要将 16 个以上的位用于类事件标识。这种格式将 7 位分配给集合标识，将 24 位分配给类事件标识。对 OSF 指定格式 A 事件类编号，其集合标识为零 (0)。即，在最重要的字节中，OSF 使用的所有事件类编号都是一个零。

格式 B 可由中等组织使用，它们需要将 8 至 16 位用于类事件标识。

格式 C 可由小组织使用，它们需要将 8 个以下的位用于类事件标识。

格式 D 不受 OSF 管理，可以在单元内部自由使用它。这些事件类编号在单元之间不能是唯一的，安装在多个单元中的应用程序服务器不应该使用它。

格式 E 保留供以后使用。

最重要位为 110 的编号（即，0xC0000000 到 0xDFFFFFFF）保留供单元内部本地使用。

图76 举例说明了事件类编号格式（类事件标识标注为 『event-id』）。

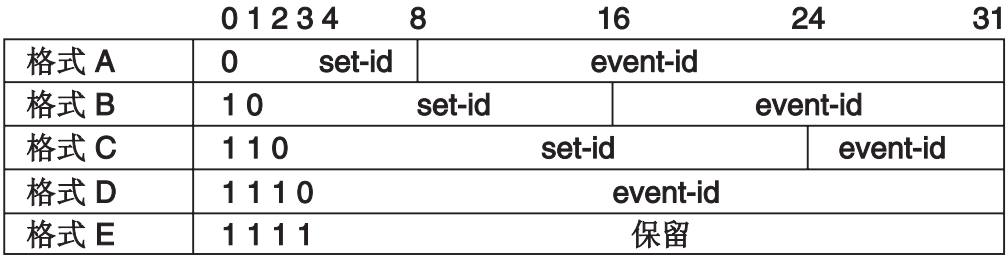


图 76. 事件类编号格式

单元管理员负责管理和分配本地事件类编号及其名称。

过滤器

一旦在应用程序服务器中标识并放置了代码点，与该代码点对应的所有审计事件都将被记录到一个审计跟踪文件中，而不考虑这些审计事件的结果。然而，记录所有情况下的所有审计事件可能既不实际也无必要。过滤器提供了一种仅当满足某些条件时才记录审计记录的方法。管理员可以使用 DCE 控制程序来定义过滤器。

过滤器由指定这些条件的过滤器向导组成。过滤器向导还指定当符合条件（结果）时要采取的操作。

过滤器回答下列问题:
- 审计对象
- 将审计什么事件？
- 在写下审计记录之前这些事件的结果应该是什么？
- 是否在审计跟踪文件中记录审计记录或（和）在系统控制台上显示它？

例如，对于银行服务器程序，可以将下列条件强加在写下审计记录之前:

『记录 DCE 单元中的所有客户（审计对象）因遇到拒绝访问（事件的结果）而导致提取事务失败（审计事件）的审计记录。』

过滤器主题身份

过滤器与一个过滤器主题相关，该主题指示过滤器的应用*对象*。过滤器主题是引发事件的分布式应用程序的客户。过滤器主题有两个部分: 过滤器类型和键。

有八种过滤器类型:
- **principal**--本地单元中的 DCE 主体。
- **foreign_principal**--外部单元中的 DCE 主体。
- **group**--本地单元中的 DCE 组。
- **foreign_group**--外部单元中的 DCE 组。
- **cell**--网络中的 DCE 单元。
- **cell_overridable**--网络中的 DCE 单元。这种类型可以被更特定的过滤器类型覆盖。

- **world**--分布式应用程序的所有客户机。
- **world_overridable**--分布式应用程序的所有客户机。这种类型可以被更特定的过滤器类型覆盖。

键是 **principal**、**foreign_principal**、**group**、**foreign_group**、 **cell** 和 **cell_overridable** 过滤器类型的特定名称。 **world** 和 **world_overridable** 过滤器类型没有键。

过滤器向导

过滤器包含一个或多个向导。过滤器向导包含三个元素：审计条件、审计操作和事件类。

审计条件指定在审计记录写入审计跟踪之前需要的事件结果。这些结果不是互斥的。审计条件有:

- **success**--仅当事件成功时记录。
- **failure**--仅当事件失败时记录。
- **denial**--仅当由于访问遭到拒绝而使事件失败时记录。

审计操作指定在哪里写下审计记录。 审计操作有:

- **alarm**--在系统控制台上显示审计记录。
- **log**--在审计守护程序中记录审计记录或直接写入审计跟踪文件。
- **ems**--将事件提供给 EMS 服务器，然后将它提供给已正确注册的 EMS 消费者。

这些审计操作不是互斥的。

现在，除了 **log** 和 **alarm** 操作外，过滤器向导中的**所有**操作还包括 **ems** 操作。如果在 DCE 的先前版本中创建了审计过滤器并在迁移到 DCE 3.2 时指定**所有**操作，则操作只包括 **log** 和 **alarm** 操作，而不包括 **ems**。

当创建或修改审计过滤器时，不要在包括任何 DCE 核心服务（如安全性或 cds）的审计事件的过滤器向导中指定 **ems** 操作。如果使用审计过滤，这会使 DCE 核心服务挂起。

第四种操作是 **none**，仅当由自身使用时才有意义。这个操作可用于覆盖特定主体、组或单元的可覆盖过滤器。

过滤器向导的第三个元素指定将应用过滤器的一个或多个事件类（对于特定过滤器主题身份）。

过滤器向导例子

下面是具有两个向导的过滤器例子:

```
{principal_cell_admin}
{{dce_sec_authent dce_sec_server} success log}
{dce_sec_authent {failure denial} {log alarm}}
```

这是键为 **cell_admin** 的主体过滤器。向导 1 指定: 如果用户是 **cell_admin** 并且操作是成功的，则将为事件类 **dce_sec_authent**（与 “DCE” 安全性认证相关的事件） 或 **dce_sec_server**（与 “DCE 安全性” 副本相关的事件）中的任何事件记录审计记录。向导 2 指定: 为结果是拒绝访问或未指明失败原因的用户 **cell_admin** 记录 **dce_sec_authent** 中的任何事件并在控制台上显示它。

过滤器规则

过滤器规则用于解析不同过滤器的重叠向导。有两种过滤器规则: 覆盖和高水位标记。

在覆盖规则中，更特定的过滤器会使可覆盖的过滤器（即，**cell_overridable** 和 **world_overridable** 类型）失效。覆盖规则用作一种允许补充过滤器的机制。主体或组的过滤器比单元或 world 的过滤器更明确。

高水位标记规则应用于覆盖规则之后。如果多个过滤器可应用于一个客户机，则应用这些过滤器指定的操作联合（log 或 alarm）。

如果客户机的主体、组或单元身份与过滤器的键匹配，则过滤器可应用于客户机。**world** 和 **world_overridable** 过滤器没有键并且应用于所有客户机。如果有多个过滤器可应用于一个客户机，则执行这些过滤器指定的操作联合（**log**、**alarm** 或 **ems**）。

使用过滤器规则例子

在以下方案中描述可覆盖的过滤器使用:

公司（单元）X 的 Alice 负责激活某些操作（事件类 **critical_transactions**）。公司中的其它主体也经授权激活相同的操作，但有某些条件限制; 例如，当 Alice 没空时。不管事件结果（即，审计条件= 全部）是什么或者是谁激活这些操作，系统管理员都要记录审计记录。如果激活者不是 Alice，则管理员还要生成一个警报。由下列两个过滤器实现这个规范:

```
Filter 1:
{principal Alice}
{critical_transactions all log}
Filter 2:
{cell_overridable X}
{critical_transactions all {log alarm}}
```

当 Alice 调用 **critical_transactions** 事件类中的事件时，主体过滤器 (filter 1) 是可应用的，因为它的键与 Alice 的身份匹配。主体过滤器比单元过滤器更明确。尽管单元过滤器 (filter 2) 也可应用于 Alice （Alice 属于单元 X），但主体过滤器覆盖它，因为单元过滤器是可覆盖的。对于公司（单元） X 中的其它主体，唯一可应用的过滤器是单元过滤器 (filter 2)。因此，这些相同事件将使得审计记录被记录下来，并且还产生一个警报。

不可覆盖的 world 和单元过滤器也是有用的。如果没有它们，例如，为了使单元范围的过滤器作用于整个单元，管理员必须删除单元的所有组过滤器和主体过滤器。（当猜想单元是安全性问题的起因时，系统管理员可能要引入一个不可覆盖的单元过滤器 *temporary*。）

第439页的图77 说明了不同类型过滤器之间的覆盖关系。从过滤器类型 X 到过滤器类型 Y 的箭头意味着 X 覆盖 Y。

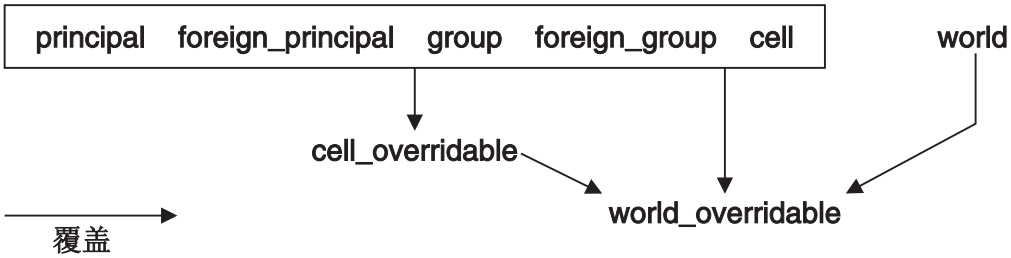


图 77. 过滤器类型之间的覆盖关系

通常，定义 DCE 组的目的是为了授予访问许可权。组过滤器指定*审计使用组特权的意图*，而不是指定*审计属于组的主体*。即，组过滤器将不影响组成员主体的审计，除非主体有意要使用组的特权（通过包括 PAC 中的组）。因为组过滤器被定义为审计使用组特权的意图，所以它们与其它过滤器无关，并且不可覆盖。

审计跟踪文件

审计跟踪文件包含审计客户机写下的所有审计记录。可以指定*中央审计跟踪文件*或*本地审计跟踪文件*。

中央审计跟踪文件由审计守护程序在启动时创建。缺省情况下，如果 **dce_aud_open( )** 功能没有为审计跟踪文件指定名称，那么所有的审计记录都将发送到审计守护程序，该程序将它们存储在中央审计跟踪文件中。

如果用跟踪文件名称来调用 **dce_aud_open(f)** 功能，则这个名称成为至本地审计跟踪文件的路径名，并且所有审计记录都将发送到该文件中。

DCE 审计中的管理和编程

许多“DCE 审计服务”管理任务与应用程序员执行的任务相关。要理解这些管理任务，您应该熟悉“DCE 审计服务”的某些编程知识。本部分描述一个典型的“DCE 审计服务”编程和管理方案及其任务。

银行服务器例子说明这个方案。

程序员任务

应用程序员使用 DCE 审计 API 来启用应用程序服务器程序中的审计。明确地讲，程序员执行下列任务:

1. 在应用程序服务器程序中标识对应于审计事件的代码点。

   例如，银行服务器程序可以有这些功能： **acct_open( )**、**acct_close( )**、**acct_withdraw( )**、 **acct_deposit( )** 和 **acct_transfer( )**。可以将其中的每个功能指定为一个代码点，就是说，它们都有可能是可以记录的审计事件（取决于过滤器）：

   ```
   acct_open( )              /* first code point */
   acct_close( )             /* second code point */
   acct_withdraw( )          /* third code point */
   acct_deposit( )           /* fourth code point */
   acct_transfer( )          /* fifth code point */
   ```

2. 为每个代码点指定一个事件编号。**dce_aud_open( )** API 和 **dce_aud_start( )** API 将事件编号用作参数，前者用于打开审计跟踪，后者用于初始化代码点的审计记录。程序员可以在服务器的头文件中定义这些事件编号。

   例如:

```
/* event number for the first code point, acct_open( ) */
#define evt_vn_bank_server_acct_open      0x01000000
/* event number for the second code point, acct_close( ) */
#define evt_vn_bank_server_acct_close     0x01000001
/* event number for the third code point, acct_withdraw( ) */
#define evt_vn_bank_server_acct_withdraw  0x01000002
/* event number for the fourth code point, acct_deposit( ) */
#define evt_vn_bank_server_acct_deposit   0x01000003
/* event number for the fifth code point, acct_transfer( ) */
#define evt_vn_bank_server_acct_transfer  0x01000004
```

3. 将对 **dce_aud_open( )** API 的调用添加到应用程序服务器的初始化例程中。这将打开审计跟踪文件。这个功能将最小的事件编号（在这种情况下是 **acct_open( )**）用作它的一个参数。例如:

```
main( )
/* evt_vn_bank_server_acct_open is the lowest event number */
dce_aud_open(aud_c_trl_open_write, description,
              evt_vn_bank_server_acct_open,
              5, &audit_trail, &status);
```

4. 将审计事件记录功能添加到应用程序服务器代码中的每个代码点。这些功能在每个代码点执行以下任务:

   - 通过使用 **dce_aud_start( )** API 初始化审计记录。这个功能将事件编号指定给表示事件的代码点。这样，这个功能将事件编号用作它的一个参数。
   - 通过使用 **dce_aud_put_ev_info( )** API，将特定于事件的信息添加到审计记录。
   - 使用 **dce_aud_commit( )** API 来落实审计记录。这个功能将审计记录写入审计跟踪文件。

   下面的例子说明如何在银行服务器程序的代码点上使用这些 API（这不是完整例子）:

```
acct_open( )     /* first code point */
/* Uses the event number for acct_open( ), */
/*                              evt_vn_bank_server_acct_open */
dce_aud_start(evt_vn_bank_server_acct_open,
              binding,options,outcome,&ard, &status);
 if (ard) /* If events need to be logged */
      dce_aud_put_ev_info(ard,info,&status);
 if (ard) /* If events were logged */
      dce_aud_commit(at,ard,options,format,&outcome,&status);
acct_close( )   /* second code point */
/* Uses the event number for acct_close( ), */
/*                              evt_vn_bank_server_acct_close */
dce_aud_start(evt_vn_bank_server_acct_close,
              binding,options,outcome,&ard, &status);
 if (ard) /* If events need to be logged */
      dce_aud_put_ev_info(ard,info,&status);
 if (ard) /* If events were logged */
      dce_aud_commit(at,ard,options,format,&outcome,&status);
```

5. 使用主服务器例程中的 **dce_aud_close( )** API，以便在服务器关闭时关闭审计跟踪文件。例如:

```
dce_aud_close(audit_trail, &status);
```

管理员任务

管理员使用表示审计客户机应用程序服务器程序中不同代码点的事件编号并以下列方式来创建事件类文件和过滤器向导:

1. 管理员从应用程序服务器程序员那里获得代码点（表示每个审计事件）的事件编号。在我们的例子中，为这些代码点指定了下列事件编号:

   acct_open( )
   0x01000000
   acct_close( )
   0x01000001
   acct_withdraw( )
   0x01000002
   acct_deposit( )
   0x01000003
   acct_transfer( )
   0x01000004

   （注意，事件编号应该是完全连续的。即，不允许丢失序列成员。）

2. 管理员决定创建两个事件类: 由 **acct_open( )** 和 **acct_close( )** 组成的 **account_creation_operations** 类，以及由 **acct_withdraw( )**、**acct_deposit( )** 和 **acct_transfer( )** 组成的 **account_balance_operations** 类。管理员将事件类 **account_creation_operations** 的事件类编号指定为 0xC0000006。将事件类 **account_balance_operations** 的事件类编号指定为 0xC0000007。

   要创建事件类，管理员创建并编辑两个文件，为每个事件类创建一个文件。其中各个文件的名称将与各个文件所表示的事件类的名称相同。每个文件将包含各个事件类中的事件数。

   名称为 **account_creation_operations** 的文件编辑如下（以 # 井号开头的行是注释行）:

   ```
   # Event class number of account_creation_operations
   ECN = 0xC0000006
   # Event number of acct_open( )
   0xC1000000
   # Event number of acct_close( )
   0xC1000001
   ```

   名称为 **account_balance_operations** 的文件编辑如下:

   ```
   # Event class number of account_balance_operations
   ECN = 0xC0000007
   # Event number of acct_withdraw( )
   0xC1000002
   # Event number of acct_deposit( )
   0xC1000003
   # Event number of acct_transfer( )
   0xC1000004
   ```

   管理员将两个文件都存储在 *dcelocal***/etc/audit/ec** 目录中。

3. 管理员决定创建两个过滤器: 一个用于单元（针对于单元 **/.:/torolabcell**）内部的所有用户，另一个用于所有其它用户。

   单元内部所有用户的过滤器有下列向导:

   - 只审计事件类 **account_balance_operations** 中的事件，视下一个条件而定。
   - 仅当由于拒绝访问而使该事件类中的操作失败时，才写下审计记录。

- 如果实现第一个条件，就只在审计跟踪文件中写下审计记录。
- 然后，管理员使用 DCE 控制程序的 **audfilter create** 命令来创建这个过滤器:

```
dcecp> audfilter create  {cell /.../torolabcell} -attribute \
> {account_balance_operations denial log}
dcecp>
```

所有其它用户的过滤器都有以下向导:

- 审计两个事件类中的事件，视下一个条件而定。
- 如果该事件类中的操作成功、失败，或由拒绝访问而导致失败，写下审计记录。
- 将审计记录同时写入审计跟踪文件和控制台。

下面是创建这个过滤器的 **dcecp** 会话:

```
dcecp> audfilter create  world -attribute \
> {{account_balance_operations account_creation_operations} {alarm log} all}
dcecp>
```

第443页的『第45章 DCE 审计服务管理任务』 提供有关 DCE 控制程序的 **audfilter create** 命令的详细信息。

第45章 DCE 审计服务管理任务

本章概述了开始使用 DCE 审计的基本步骤。另外本章还描述了以下对 “DCE 审计服务” 执行的管理任务:

- 设置 DCE 审计环境变量。
- 启动（和停止）DCE 审计守护程序。
- 控制对 DCE 审计守护程序的访问。
- 创建和维护事件类，以在逻辑上将一组审计事件分组。通过编辑事件类文件创建事件类。
- 创建和维护设置在审计跟踪文件中记录审计事件的标准的过滤器。
- 启用和禁用 DCE 审计守护程序的审计记录服务。
- 修改和查询 DCE 审计守护程序的属性。
- 控制和显示审计跟踪文件。
- 使用 DCE 可服务性路由文件。

在 AIX 和 Solaris 上使用 DCE 审计

以下步骤用于帮助 DCE 单元管理员快速和轻易地开始在 AIX 和 Solaris 上使用 DCE 审计。它处理与 DCE 服务器一起提供的缺省事件，并不用于描述如何在您自己的应用程序中使用 DCE 审计。虽然概念一般适用于 DCE 审计，但审计 DCE 安全性服务器守护程序是主要的重点。

配置 auditd 守护程序

运行以下命令来配置 DCE auditd 守护程序:

```
config.dce audit
```

至少应该在配置了安全性服务器的主机上配置 auditd 守护程序。可以使用以下 dcecp 命令来显示本地审计守护程序的状态:

```
dcecp -c aud show
```

DCE 审计客户机使用 auditd 守护程序的服务来执行审计，客户机通知该审计守护程序以审计事件，该审计守护程序将审计记录写回审计跟踪。例如，dtsd 时间守护程序和 secd 安全性服务器守护程序是审计客户机。

停止和重新启动 DCE 服务器

必须设置 **DCEAUDITON** 环境变量来重新启动 DCE 服务器，以使它们能够使用主机上的审计服务。

```
stop.dce
export DCEAUDITON=1
start.dce
```

Start.dce 将在启动任何 DCE 守护程序 之前在内部设置 **DCEAUDITON = 1**。

收集审计记录

DCE 服务器现在正在收集审计记录并将它们存储在审计跟踪中。审计子系统使用过滤器来确定要创建哪个审计事件的审计记录以及要将审计跟踪写入哪个事件。缺省地，审计使用“审计所有”过滤器来收集审计记录（它在配置审计时创建）。

安全性服务器事件的审计跟踪存储在

```
/opt/dcelocal/var/security/sec_audit_trail
```

显示审计跟踪

使用审计的目标是显示审计跟踪。检查审计跟踪通过显示发生了什么重大事件、何时发生以及由谁来执行的，从而帮助改进您的 DCE 单元的安全性。

在审计跟踪文件中使用以下 dcecp 命令来在审计跟踪内显示审计记录:

```
dcecp> audtrail show /opt/dcelocal/var/security/sec_audit_trail
```

使用基于名称的授权的登录尝试审计和 RPC 服务器功能方面的更改

在 DCE 以前的版本中，如果 audited 事件的授权机制只依赖名称，则当使用无效名称时（例如，使用了无效 DCE 主体的字符串），将永远不会正确审计事件。

在这个发行版中，当审计任何只依赖名称实现安全性（基于名称授权的 RPC，预认证）的事件并向事件授予非 DCE 主体名称时，将正确审计事件。如果所用的名称是有效的 DCE 主体名称，则将把主体的 UUID 作为*客户机 UUID*存储在审计记录中。如果所提供的名称无效，则将把客户机的 UUID 全部设置成零，授权状态将为 **rpc_c_authz_name**（由 **dce_aud_print( )** 和 **dcecp audtrail show( )** 显示为*用名称授权*），名称将存储为审计记录中的第一个特定于事件的项目。

这不是基于名称授权的缺省行为，但是它已经对 AS_Request、TGS_TicketReq、TGS_RenewReq 和 TGS_ValidateReq 事件启用。一些示例:

- 用户用 **dce_login** 和一个非 DCE 主体登录，给出一个不正确的第三方协议口令。那个事件 (AS_Request 0x101) 的审计记录将包含故障输出、**rpc_c_authz_name** 的授权状态，和零客户机 UUID，而且唯一特定于事件的项将是用户作为主体授予 **dce_login** 的字符串。
- 用户使用 **dce_login** 和一个有效 DCE 主体登录，给出一个不正确的第三方协议口令。那个事件 (AS_Request 0x101) 的审计记录将包含拒绝输出、**rpc_c_authz_name** 的授权状态和标识授予 **dce_login** 的主体名称的客户机 UUID。
- 用户使用 **dce_login** 和 Entrust 用户概要的名称登录，给出不正确的那个概要的 Entrust 口令短语。审计记录将包含拒绝输出和 **rpc_c_authz_name** 的授权状态。如果 Entrust 概要名称与有效的 DCE 概要名称不匹配，则客户机 UUID 将全部为零，并且概要名称（授予 **dce_login**）将是审计记录中唯一特定于事件的项。
- 用户使用 **dce_login** 和 Entrust 用户概要的名称登录，给出那个概要的 Entrust 口令短语。那个事件 (AS_Request 0x101) 的审计记录将包含成功输出和 **rpc_c_authz_name** 的授权状态。如果 Entrust 概要名称不与有效的 DCE 概要名称匹配，则客户机 UUID 将全部为零，并且提供给 **dce_login** 的概要名称将是审计记录中唯一特定于事件的项。

存储在审计记录中的名称是在尝试登录时用户提供的名称。如果该名称恰巧是有效的 DCE 主体名称，则将存储主体的 UUID。否则，该名称将出现在特定于事件的信息中。

用 PKC 登录，如果 Entrust 用户概要名称与任何 DCE 主体名称相同，则那个主体将在审计记录中作为客户机 UUID 存储，并且将没有特定于事件的信息。

设置 DCE 审计环境变量

有三个与 DCE 审计服务的操作相关的环境变量。应该在运行应用服务器（也就是 DCE 审计客户机）之前设置 DCE 审计环境变量。有以下这些环境变量:

- **DCEAUDITOFF**--如果在启动应用程序时声明该变量，则将关闭审计。缺省地，不定义该变量。
- **DCEAUDITFILTERON**--如果在启动应用程序时声明该变量，则启用过滤。缺省地，不声明该变量，也就是说，没有过滤并且记录所有审计事件。
- **DCEAUDITTRAILSIZE**--设置审计跟踪的最大尺寸。
- **DCEAUDITWRAP**--跟踪将在达到最大尺寸时回绕。

启动审计守护程序

DCE 审计服务不是分布式应用程序。即使客户机应用程序正在使用审计服务，审计守护程序 (**auditd**) 也不需要在所有的 DCE 主机上运行。如果审计日志将存储在中央跟踪文件或者如果过滤器将安装在主机，则审计守护程序只需要运行在一台主机上。这是因为审计守护程序控制对中央跟踪文件的访问并管理审计过滤器。然而，既然 DTS 守护程序和安全性服务器守护程序是审计客户机，您可能要考虑在这些服务器所运行单元中的所有主机上运行审计守护程序。

必须是 **root** 用户才能启动审计守护程序。

使用以下命令来启动审计守护程序:

```
start.dce audit
```

该命令使用影响守护程序行为的标志。有关这些标志的详细信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference*。

控制对审计守护程序的访问

必须控制对审计守护程序的访问，以防止未经授权的应用程序服务器（审计客户机）使用它。如果未经授权的服务器能够记录其审计记录，那么审计存储空间将耗尽。

通过使用 **dcecp** 编辑审计守护程序对象 **/.:/hosts/***hostname***/audit-server** 的 ACL 来控制对审计守护程序的访问。

DCE Audit 服务支持的 DCE 许可权

DCE Audit 服务支持以下可用来定义审计守护程序 ACL 的 DCE 许可权:

r 读许可权。允许主体读过滤器。

w 写许可权。允许主体修改过滤器。

c 控制许可权。允许主体控制审计守护程序。这包括启用和禁用记录服务的能力，以及修改审计守护程序的 ACL 的能力.

l 记录许可权。允许主体在审计跟踪文件中写跟踪记录。

审计守护程序的初始 ACL

主机审计守护程序的初始 ACL 包含以下项:

```
{unauthenticated -r--}
{user hosts/nodoz/self crwl}
{group subsys/dce/audit-admin crwl}
{any_other -r--}
```

第一项允许任何未经授权的用户以只**读**方式访问过滤器。第二项允许主机主体 (**hosts/**<*hostname*>**/self**) 查询和修改过滤器，控制审计守护程序以及写入审计跟踪文件。第三项允许组 **subsys/dce/audit-admin** 的成员具有与主机主体同样的访问权。最后一项只允许所有其它主体具有对过滤器的**读**访问权。您可以通过使用 **dcecp** 来修改这个 ACL，以适合您的安全性需要。

为审计客户机和管理员授予许可权

通过使用 **dcecp**，您可以向将为审计客户机授予审计跟踪文件**记录**许可权的审计守护程序的 ACL 中添加项。可以创建一个 DCE 安全性组，该组由主机上被授权生成审计记录的服务器组成。例如:

group/hosts/<*hostname*>**/audit-clients**

为这个组授予对审计守护程序的**记录**许可权。例如:

```
dcecp> acl modify /.:/hosts/machine1/audit-server \
> -add {group hosts/machine1/audit-clients l}
dcecp>
```

然后，所有审计客户机都可以成为该组的成员，并继承它对审计守护程序的许可权。

还必须添加 ACL 项，以为指定的管理员授予对审计守护程序的读、查询和控制许可权。例如，对于管理员的组 **group/hosts/machine1/audit-admin**:

```
dcecp> acl modify /.:/hosts/machine1/audit-server \
> -add {group hosts/machine1/audit-admin rwc}
dcecp>
```

定义事件类

可以把单独的审计事件分组在一起，以形成事件类。事件类提供一种进行逻辑分组和使用单一值进行选择的有效机制。

DCE 审计事件类是可配置的。可以添加或除去现有事件类的事件或定义新的事件类。

定义本地事件类的能力在简化多 DCE 应用程序中管理审计服务方面很有用。管理员可以设计他们自己的反映其安全性需要的审计事件类，并跟踪存储器资源约束。

还可以创建临时的事件类来跟踪安全性违例。

定义事件类的步骤

要定义事件类，请执行以下步骤:

1. 从单元管理员处获得事件类的事件类号。 OSF 应该已经为您的组织分配了一定范围的事件类号。如果没有分配，则与 OSF 联系。
2. 在 *dcelocal***/etc/audit/ec** 目录中创建一个事件类文件。如下所示编辑文件:
   a. 通过以如下格式添加一行来声明*事件类号* (ECN):

```
ECN=_event_class_number
```

主机上所有事件类的 ECN 都必须唯一。

b. 可选地，您可以在该文件中添加一个*服务器事件前缀* (SEP) 行。 SEP 行包含每个服务器的事件号前缀。事件号前缀是每个服务器中最低的事件号。 SEP 行有以下格式:

```
SEP=_event_number1 event_number2 event_number3 ...
```

可以将 SEP 行放在文件中的任何位置。SEP 行通过跳过无关的事件类文件来加快审计客户机的扫描速度。

c. 从应用程序获得要在事件类中包括的代码点的事件号。

d. 添加与要在事件文件中包括的事件相对应的事件号，每行一个号。

在事件类文件中，忽略空行，通过在注释文本之前加上一个 # （数字符号）来指定注释。

示例事件类文件

以下是名为 **ec_local_cell_critical_events** 的样本事件类文件:

```
ECN = 0xC0000005
# Server Event Number Prefixes
# 0x000001 Security Service Events
# 0x000002 Time Service Events
# 0x000003 Audit Service Events
SEP = 0x00000100 0x00000200 0x00000300
# Security Service Critical Events
# evt_osf_dce_rs_properties_set_info (sets registry properties)
0x0000011f
# evt_osf_dce_rs_policy_set_info (sets registry policy)
0x00000121
# evt_osf_dce_rs_rep_admin_stop (stops the registry service)
0x00000127
# evet_osf_dce_rs_rep_admin_mkey (changes master key)
0x00000129
# Time Service Critical Events
# evt_osf_dce_dts_create (creates a server or a clerk)
0x00000201
# evt_osf_dce_dts_delete (deletes a server or a clerk)
0x00000202
# evt_osf_dce_dts_enable (enables the time service)
0x00000203
# evt_osf_dce_dts_disable (disables the time service)
0x00000204
# Audit Service Critical Events
# evt_osf_dce_aud_enable (enables audit-record logging service)
0x00000301
# evt_osf_dce_aud_disable (disables audit-record logging service)
0x00000302
# evt_osf_dce_aud_stop (terminates the execution of the audit daemon)
0x00000303
```

创建和维护过滤器

启动了审计守护程序并创建了事件类文件之后，可以运行 **dcecp** 来创建、修改或显示由审计守护程序维护的过滤器。使用 **audfilter create**、**audfilter modify** 和 **audfilter delete** 命令来创建、修改和删除过滤器。使用 **audfilter catalog** 和 **audfilter show** 命令来显示现有过滤器。

创建过滤器

以下是用于创建过滤器 **audfilter create** 命令的示例:

```
dcecp> audfilter create  {group trust} \
> -attribute {ec_local_bank_audit denial log}
dcecp>
```

示例命令指定，为本地单元中名为 **trust** 的 DCE 组创建过滤器类型 **group**。

需要 **-attribute** 选项。传递给该选项的自变量是过滤器操作指示或操作指示列表。每个过滤器操作指示都由三个元素组成: 一个*事件类名称*或名称列表、一个*审计条件*或条件列表和一个*审计操作*或操作列表。

事件类名称与您为之创建过滤器的事件类文件相对应。

审计条件是审计事件所必需的条件。有效条件是**成功**、**拒绝**、**失败**、**暂挂**和**全部**。

审计操作是在生成的事件与指定的审计条件匹配时采取的操作。有效的操作是**无**、**日志**、**警报**、**ems** 和**全部**。

修改过滤器

可以通过添加或删除一个或多个过滤器操作指示来修改现有的审计过滤器。以下是用于修改现有过滤器的 **dcecp** 命令的示例:

```
dcecp> audfilter modify world -add {Monetary_Transfers denial log}
dcecp>
```

该示例命令将事件类为 **Monetary_Transfers**、审计条件为**拒绝**并且审计操作为**日志** 的操作指示添加到现有过滤器类型**审计所有**中。请注意，过滤器类型**审计所有**不采用密钥。

DCE 控制程序不使用逗号。以标准的 **dcecp** 列表格式指定多个操作指示和多个过滤器: **{x y}** 用于单一自变量，或者 **{{x y} {a b}}** 用于多个自变量。

为了执行 **audfilter modify** 命令，必须有审计守护程序的 ACL 的写 (**w**) 权限。

删除过滤器

可以通过使用 **audfilter delete** 命令删除 DCE 客户机的一个或多个审计过滤器。下面是 **audfilter delete** 命令的示例:

```
dcecp> audfilter delete  {foreign_principal/.../foreign_cell_name/jedwards}
dcecp>
```

示例命令删除外部单元 **/.../foreign_cell_name** 中 DCE 主体 **jedwards** 的审计过滤器。

可以在 **audfilter delete** 命令中指定多个要操作的过滤器。在前面的修改过滤器示例中，当删除多个过滤器时，必须使用标准的 **dcecp** 语法。

为了执行该命令，必须有审计守护程序的 ACL 的写 (**w**) 许可权。

缺省过滤器

在配置主机期间（使用 **dce_config**），执行以下 **audfilter create** 命令（使用 **dcecp**）来创建安全性守护程序、DTS 守护程序和审计守护程序的过滤器:

```
audfilter create world -at {dce_sec_modify success log}
audfilter create world -at {dce_sec_modify {failure denial} all}
audfilter create world -at {dce_sec_server success log}
audfilter create world -at {dce_sec_server {failure denial} all}
audfilter create world -at {dce_sec_authent {failure denial} all}
audfilter create world -at {dce_sec_query denial all}
audfilter create world -at {dce_dts_mgt_modify success log}
audfilter create world -at {dce_dts_mgt_modify {failure denial} all}
audfilter create world -at {dce_dts_mgt_query {failure denial} all}
audfilter create world -at {dce_audit_admin_modify success log}
audfilter create world -at {dce_audit_admin_modify {failure denial} all}
audfilter create world -at {dce_audit_filter_modify success log}
audfilter create world -at {dce_audit_filter_modify {failure denial}
all}
audfilter create world -at {dce_audit_admin_query {failure denial} all}
audfilter create world -at {dce_audit_filter_query {failure denial}
all}
```

启用审计过滤器

如果要启用审计过滤器，必须首先设置 **DCEAUDITFILTERON** 环境变量。必须在启动服务器（即审计客户机）之前设置该变量。

除去更新绑定文件

如果服务器（审计客户机）在启用了过滤器（即设置了 **DCEAUDITFILTERON**）的情况下运行，则**libaudit**（它链接到服务器）获得服务器的绑定信息并将它存储在以下位置:

/opt/dcelocal/var/audit/client/*pid-of-server***/update_binding_file**

where *pid-of-server* is the process ID of the server.

如果服务器异常终止，则必须手工除去该文件。如果不除去该文件，将在下一次用 **DCEAUDITFILTERON** 重新启动服务器时接收到一条错误消息。该消息指出: 审计守护程序无法将过滤器更新通知给审计客户机:

```
unable to inform process
/opt/dcelocal/var/audit/client/pid-of-server/update_binding_file
about esl update.
```

您可以通过查看哪些服务器在运行（例如，使用 **ps -e**）并比较其进程标识与更新绑定文件的路径名，来检查是否有旧的更新绑定文件。因为这些文件包含的路径名包含一个 *pid-of-server* 组件，因此可以确定哪些文件对应于不存在的服务器。

必须除去绑定信息文件和包含它的目录 (*pid-of-server*)。

审计跟踪缓冲

在将审计跟踪数据写入磁盘之前，操作系统缓冲审计跟踪。由于这个原因，在将数据从缓冲区写入磁盘之前，审计跟踪文件的增长将不明显。

启用和禁用审计记录服务

使用 **dcecp** 来启用或禁用审计守护程序审计记录的记录服务。 **aud enable** 命令启用记录服务，**aud disable** 命令禁用它。

您可能要在审计跟踪文件变得过大时禁用记录服务，在备份并回绕了审计跟踪（使用 **aud rewind** 命令）之后重新起用它。

使用 enable 或 disable 命令将启用或禁用将审计记录写到中央审计跟踪文件的功能。象安全性服务器和时间服务器这样的应用程序使用它们自己的审计跟踪文件，并且不受使用 enable 或 disable 的影响。

aud stop 命令停止审计守护程序。

修改和查询审计守护程序属性

DCE 审计守护程序有两个与审计跟踪文件相关的属性:

- **stostrategy**--指定当审计跟踪文件达到其极限时的存储器策略。可以指定以下存储器策略之一:

  save 如果达到了指定的跟踪大小，则审计守护程序将中央跟踪文件存入一个新文件（将它重命名成其初始名称后面加上时间戳记）。然后，审计守护程序删除初始跟踪文件的内容，并继续从该文件的开始处进行审计。这是 **stostrategy** 的缺省值。

  wrap 当中央跟踪达到其最大尺寸时（由 **dce_aud_set_trail_size( )** 或环境变量 **DCEAUDITTRAILSIZE** 设置），审计守护程序开始在文件的开始处覆盖审计记录，但不删除文件。当把新记录方入跟踪时，覆盖跟踪内最早的审计记录。

- **state**--指出审计守护程序是否为客户机发出的审计记录的记录请求服务。该属性可能的值是 **enabled**（缺省值）或 **disabled**。

可以使用 **dcecp** 来查看这些设置的值，如下所示:

```
dcecp> aud show
{state enabled}
{stostrategy save}
dcecp>
```

使用 **aud modify** 命令来更改这些属性。

控制和显示审计跟踪

审计守护程序 将审计客户机发出的审计记录写入审计跟踪文件。如果不带任何自变量启动审计守护程序，则所用的缺省审计跟踪文件是 *dcelocal*/**var/audit/adm/central_trail**。还可以通过在启动守护程序时使用 **auditd** 命令的 **-t** 选项来将审计跟踪重定向到另一个文件；传递给 **-t** 选项的 *trail* 自变量指定要将日志写入的文件的路径名。

显示审计跟踪文件

使用 **dcecp audtrail show** 命令来检查审计跟踪文件的内容。可以显示中央或本地审计跟踪文件的内容。

例如，可以使用以下命令来查看审计跟踪文件 **central_trail** 的内容:

```
dcecp> audtrail show /opt/dcelocal/var/audit/adm/central_trail
--- Event Record number 31 --
o Event Information:
  - Event Number:                0x113 /* 275 */
  - Event Name:                  LOGIN_GetInfo
  - Event Outcome:               success
o Server:                        /.:/hosts/zinfandel.austin.ibm.com
o Client:                        /.../budokan_3/hosts/zinfandel.austin.ibm.com/self
o Number of groups:              0
o Authorization Status:          Authorized with a pac
o Date and Time recorded:        1999-01-19-11:48:40.538-06:00I----
o 1 Event(s) specific:
  - item number 1                char string hosts/zinfandel.austin.ibm.com/self
--- End of Event record number 31 --
--- Event Record number 34 --
o Event Information:
   - Event Number:               0x113 /* 275 */
   - Event Name:                 LOGIN_GetInfo
   - Event Outcome:              success
o Server:                        /.:/hosts/zinfandel.austin.ibm.com
o Client:                        /.../budokan_3/hosts/zinfandel.austin.ibm.com/self
o Number of groups:              0
o Authorization Status:          Authorized with a pac
o Date and Time recorded:        1999-01-19-11:48:40.873-06:00I----
o 1 Event(s) specific:
  - item number 1                char string hosts/zinfandel.austin.ibm.com/self
--- End of Event record number 34 --
```

如果希望将审计跟踪数据放入文件，而不是在屏幕上显示，则在 **audtrail show** 命令行包括 **-to** 选项。该选项将审计跟踪文件的内容打印到指定的文件名。对于大型跟踪文件，强烈建议使用该选项。

控制审计跟踪大小

缺省地，审计跟踪文件限制在 2 MB 大小。当审计服务检测到该跟踪文件的大小超过这个值时，它关闭该文件，通过使用时间戳记信息为该文件创建一个新的唯一名称，然后用初始名称打开新的跟踪文件。然后，它继续将新的跟踪日志写入这个文件。当这个文件的大小过大时，重复这个过程。

如要更改审计跟踪文件的大小，必须在启动使用审计服务的应用程序之前，将环境变量 **DCEAUDITTRAILSIZE** 设置成要求的大小。设置这个环境变量覆盖缺省的 2 MB 大小的限制。在服务器中，可以通过对 **dce_aud_set_trail_size_limit()** 的调用来覆盖该环境变量的值。

例如，如果希望使用 5 MB 的跟踪文件，则 **DCEAUDITTRAILSIZE** 的值应该如下:

```
DCEAUDITTRAILSIZE 5000000
```

还可以允许审计守护程序在中央跟踪文件达到其限制（缺省的 2 MB 或由 **DCEAUDITTRAILSIZE** 设置）时回绕该文件。要这样做，应该用 **-wrap** 选项启动审计守护程序:

```
auditd -wrap
```

如果就的审计记录很少有、或没有价值，而您有只想保持相对新的记录，那么您可能也想使用这个选项。

还可以使用 **auditd** 的 **-s** 选项来设置跟踪大小的限制。用这种方法设置的限制覆盖缺省的 2 MB 限制。

如果出于任何原因希望在审计跟踪达到其限制之前做一个跟踪快照，则可以使用 **dcecp aud disable** 命令来禁用记录，然后复制文件。然后可以使用 **dcecp aud rewind** 命令来回绕中央审计跟踪文件。（请注意，如果需要，可以在此时备份这个跟踪文件。但是，如果需要备份，最好让审计服务自动创建新的跟踪文件并将这些备份）。然后使用 **aud enable** 命令来重新启用审计守护程序的记录服务。

更改审计跟踪文件存储器选项

可以在运行审计守护程序时更改存储器策略选项。这只能在中央审计跟踪文件上执行。

以下示例显示如何使用 **aud modify** 命令来使审计跟踪在达到文件限制时回绕。

```
dcecp> aud modify -stostrategy wrap
dcecp>
```

这个示例命令将审计守护程序的存储器策略属性更改成 **wrap**。

第46章 Kerberos 与 DCE 和安全远程实用程序的互操作

“DCE 安全性服务”的认证部分主要基于 Kerberos 网络认证系统的第 5 版，这在“因特网工程任务组”(IETF) RFC 1510 中有描述。这个基础在很大程度上允许“DCE 安全性服务器”作为 Kerberos V5 客户机的 Kerberos 密钥分发中心 (KDC) 操作。在 DCE 的以前版本（版本 1.2.1 和更早的版本）中，并没有正式支持或记载了这种互操作性。

本章解释 DCE 和 Kerberos V5 的互操作性特性和配置。 DCE 的当前发行版提供了以下主要功能:

- KDC 互操作性: 可以将“DCE 安全性服务”作为 Kerberos V5 的 Kerberos KDC 使用。
- 凭证高速缓存和密钥表文件的兼容性: DCE 和 Kerberos V5 应用程序能够共享凭证高速缓存和密钥表文件而不丢失数据。

KDC 互操作性

IETF RFC 1510 定义了 Kerberos V5 实现所必须的互操作性要求。尽管可以将“DCE 安全性服务器”作为 Kerberos V5 客户机的 DCE KDC 使用，但是有以下限制:

- DCE KDC 不支持的算法有以下几个:
  - DES-CBC-MD5 加密
  - CRC-32 校验和
  - DES-MAC 校验和
  - DES-MAC-K 校验和
  - DES-MD5 校验和
- 不支持跨领域通信，包括 DCE 单元和 Kerberos V5 领域之间的传递信任。
- Kerberos V5 不支持 DCE 第三方预认证。因此，如果与主体连接的 *pre_auth_req* ERA 的值为 2 (**PA-ENC-THIRD-PARTY**)，则主体无法从 DCE KDC 检索 TGT。

凭证高速缓存和密钥表文件兼容性

为了使 DCE 客户机和 Kerberos V5 客户机在同一台主机上共存，它们需要能够共享凭证和密钥。 DCE 版本 1.2.2 提供兼容的凭证和密钥表文件格式，以允许这种共存。

从 Kerberos V5 的第五个 beta 版开始，存在三种凭证高速缓存文件的格式版本和两种密钥表文件的格式版本。

不是所有的文件格式都互相兼容。例如，以版本 1 的格式写凭证高速缓存文件的老一些的 Kerberos V5 客户机将不能读版本 2 或 3 格式的凭证高速缓存。 表32 和 第454 页的表33 描述了凭证高速缓存和密钥表文件的兼容性。

表 32. 凭证高速缓存文件

发行版	Kerberos V5（版本 1）	Kerberos V5（版本 2）	Kerberos V5（版本 3）
DCE 1.0 或 Kerberos V5 beta 1	是	否	否
Kerberos V5 beta 2-3	否	是	否

表 32. 凭证高速缓存文件 (续)

发行版	Kerberos V5（版本 1）	Kerberos V5（版本 2）	Kerberos V5（版本 3）
DCE 1.1-1.2.1 或 Kerberos V5 beta 4	是	是	否
DCE 1.2.2 或 Kerberos V5 beta 5-7	是	是	是

Kerberos V5 的 Beta 2 和 Beta 3 发行版中有一个缺陷使该版本与用版本 1 格式创建的文件不兼容，这在 Beta 4 中得到修正。

表 33. 密钥表文件

发行版	Kerberos V5（版本 1）	Kerberos V5（版本 2）
DCE 1.0 或 Kerberos V5 beta 1	是	否
Kerberos V5 beta 2-3	否	是
DCE 1.1-1.2.2 或 Kerberos V5 beta 4-7	是	是

可以用 **dcecp hostvar set** 命令设置 DCE 客户机使用的凭证高速缓存和密钥表文件的格式。两个文件的缺省格式是版本 1。可以在每台主机上更改这个设置，以允许 Kerberos V5 应用程序在您的 DCE 客户机上运行并共享密钥表和凭证文件。

例如，使用以下 **dcecp** 命令将 Kerberos V5 凭证高速缓存文件的版本号格式设置成 3:

```
dcecp> hostvar set -krbccachevno 3
dcecp>
```

要将 Kerberos V5 密钥表文件的版本号格式设置成 2:

```
dcecp> hostvar set -krbktvno 2
dcecp>
```

要显示当前设置:

```
dcecp> hostvar show -krbccachevno -krbktvno
dcecp>
```

有关详细信息，请参阅 *IBM DCE Version 3.2 for AIX and Solaris: Administration Commands Reference*。要想使更改生效，停止、然后再重新启动主机上的 DCE 守护程序。

配置命令 (kerberos.dce)

kerberos.dce 命令用来使现有的 DCE 客户机和服务器可以使用 Kerberos。

注: 除了瘦客户机以外，任何用 DCE 3.1 或以后版本所带的 **config.dce** 命令配置的机器不需要运行该命令。这些步骤将自动执行。

必须在所有瘦客户机上运行这条命令，以启用 Kerberos 功能。该命令创建 **/etc/krb5.conf** 文件、**/.:/host/** 和 **/.:/ftp/** 帐户以及这些帐户的密钥表项。

1. 在现有的单元中，带 **-type admin** 选项运行该命令，以创建 **/.:/hosts/not_reg_mach_addrs**。该项只需创建一次。

2. 在要使用这项功能的单元中的每台机器上带 **-type local** 选项运行该命令。这将在 **/.:/hosts/not_reg_mach_addrs** 选项中注册每个 IP 主机地址的绑定。
3. 定期运行带 **-type admin** 选项的命令，以便为用户在 **/.:/hosts/not_reg_mach_addrs** 项之下注册的每个绑定创建主机和 FTP 帐户。
4. 在管理员完成步骤 3 之后再次在要使用这项功能的单元中每台机器上带 **-type local** 选项运行该命令。这将为这台机器的每个帐户创建密钥表项。

如果管理员愿意，可以带 **-type admin** 和 **-ip_name** 选项运行该命令。这将执行前面的第 2 步和第 3 步所执行的步骤。

DCE/Kerberos 互操作性增强

DCE 3.1 AIX 和 Solaris 版引入了以下 DCE/Kerberos 互操作性增强。

- KDC 互操作性: 可以将 DCE 安全性服务器用作 Kerberos V5 客户机的 Kerberos KDC。
- 凭证高速缓存和密钥表文件兼容性: DCE 和 Kerberos V5 应用程序可以共享凭证高速缓存和密钥表文件而不丢失数据。

其它的增强通过支持安全 AIX V4.3 远程命令（**rlogin**、**rsh**、**rcp**、**telnet** 和 **ftp**）扩展了 DCE/Kerberos 互操作性增强，并且它们可以由用户应用程序使用。这些增强没有在 *IBM DCE 版本 3.2 AIX 和 Solaris 版: 管理指南--核心组件*中记录，但是在下面有所描述。有关远程服务命令的详细信息，请参阅 AIX Version 4.3 System User's Guide 中 *Communications and Networks* 一节。

- GSSAPI 互操作性: DCE 应用程序可以使用 GSSAPI 组件来与 Kerberos V5 应用程序的 GSSAPI 组件进行组件之间的互操作。
- 对 **dce_login** and **dceunixd** 中 Kerberos 标志的支持: DCE **dce_login** 和 **dceunixd** 命令可与 Kerberos 可延期、可转发或可代理标志一起使用来获得凭证。**kinit** 可用来推迟日期。
- **k5dcelogin** 命令和 API: 应用程序可以使用 **k5dcelogin** 命令或 API 来将 Kerberos V5 凭证提升到 DCE 凭证。通过这样做，可以使用凭证来访问 DCE 对象，例如“分布式文件系统 (DFS)”文件。
- 配置和管理支持: DCE 提供了可用来配置和更新 Kerberos V5 **krb5.conf** 文件的工具，以及由安全 AIX V4.3 远程服务命令所使用的帐户。

DCE GSSAPI 组件已经得到增强，以支持以下 DCE/Kerberos 互操作性方案:

- Kerberos **gss_init_sec_context** 和 DCE **gss_accept_sec_context** 之间的组件间互操作，支持所有标志和参数。
- DCE **gss_init_sec_context** 和 Kerberos **gss_accept_sec_context** 之间的组件间互操作，除了 **GSS_C_DELEG_FLAG** 之外，支持所有标志和参数。
- Kerberos **gss_seal** 和 DCE **gss_unseal** 之间的组件间互操作，支持所有标志和参数。
- DCE **gss_seal** 和 Kerberos **gss_unseal** 之间的组件间互操作，支持所有标志和参数

为支持这些互操作方案，DCE 现在支持一种新的机制类型: **GSS_MECH_MIT_KRB5**。这种机制类型与 Kerberos V5 发行版 1 所用的缺省机制类型 **gss_mech_krb5** 具有相同的 OID 和行为。

为了向后兼容，IBM DCE V3.2 AIX 版仍然支持 DCE 早期发行版所提供的 Kerberos 机制 **GSS_C_OID_KRBV5_DES**。然而，**GSS_C_OID_KRBV5_DES** 机制类型不再与任何未经修改的 Kerberos V5 发行版互操作。

GSS_MECH_MIT_KRB5 机制类型由每一个具有机制类型参数的 GSSAPI API 支持。下面列出了这些 API，并提供了有关如何将 **GSS_MECH_MIT_KRB5** 机制类型与这些 API 一起使用的信息:

- **gss_accept_sec_context**（输出 **actual_mech_type 参数**）

  如果从使用 Kerberos V5 机制的启动程序接收到上下文，则返回 **GSS_MECH_MIT_KRB5**。

- **gss_acquire_cred**（输入 **desired_mechs 参数**）

  如果正在获得 **GSS_C_INITIATE** 或 **GSS_C_BOTH** 类型凭证（用使用 Kerberos V5 机制的接收程序初始化上下文时用到该凭证），则指定 **GSS_MECH_MIT_KRB5**。

  注: 如果正在获得 **GSS_C_ACCEPT** 类型凭证，则不使用机制类型参数。这对于所有的机制类型都正确。

- **gss_display_status**（输入 **mech_type 参数**）

  如果显示正在使用 Kerberos V5 R1 机制的上下文状态，则指定 **GSS_MECH_MIT_KRB5**。

- **gss_indicate_mechs**（输出 **mech_set 参数**）

  返回包含 **GSS_MECH_MIT_KRB5** 作为 DCE GSSAPI 所支持的机制之一的机制集。

- **gss_init_sec_context**（输入 **mech_type 参数**）

  如果正在用使用 Kerberos V5 机制的接收程序初始化上下文，则指定 **GSS_MECH_MIT_KRB**。

- **gss_inquire_cred**（输出 **mech 参数**）

  如果可以使用凭证来用 Kerberos V5 接收器初始化上下文，则返回包含 **GSS_MECH_MIT_KRB5** 的机制集。

- **gssdce_login_context_to_cred**（输入 **desired_mechs 参数**）

  如果正在获得凭证（用 Kerberos V5 CSSAPI 接收程序初始化上下文时用到该凭证），则指定 **GSS_MECH_MIT_KRB5**。（可以指定多个机制类型。）

附录A. CDS 的有效字符和命名规则

本附录讨论 CDS 接口使用的 DCE 目录服务名称的有效字符集。还说明了一些具有特殊意义的字符，并根据大小写匹配、语法和大小限制描述一些限制和规则。它不是 CDS 和 DNS 的综合引用，而是给出了记住每个服务的关键点的概述。对于 DNS 名称中有效字符的特定信息，请参阅每种技术的文档。

DCE 中对于名称的使用包含多个目录服务。例如，CDS 与任一个 DNS 交互使用找出本地单元外的名称。

注: 因为 CDS 和 DNS 都具有它们自己有效的字符集和语法规则，最好的方法是使名称简短和简单，包括对所有三个服务公共的最小字符集。建议使用字母 A 至 Z，a 至 z 和数字 0 至 9 的集合。除了使目录服务交互操作更简单之外，使用此子集还能减少用户在不同种类的硬件和软件环境中创建和使用名称遇到问题的可能性。

第458页的图78描述了 CDS 名称中有效字符的详细信息，以及 CDS 接口使用的 DNS 名称中的有效字符。

- 白色框中的字符在所有三种名称类型中都有效。
- 浅灰色框中的字符仅在 CDS 和 GDS 名称中有效。
- 深灰色框中的字符仅在 CDS 名称中有效。

图 78. CDS 和 DNS 名称中的有效字符

虽然空格在 CDS 名称中有效，但当通过 CDS 控制程序输入包含空格的 CDS 简名时，该名称仍必须包含在 『 』（引号）中。附加的接口特定规则记录在应用它们的应用中。

元字符

某些字符对目录服务具有特殊意义；这些字符称为元字符。表34 列出并说明了 CDS 和 DNS 元字符。

表 34. 元字符及其意义

目录服务	字符	含义
CDS	/	隔离名称元素（简单名称）。
	\	在需要时用在 ****（反斜杠）之前转义字符；表明下列字符不是元字符。
	,	隔离 RDN 内多个属性类型／值对（属性值断言）。
	\=	隔离属性值断言中的属性类型和值。
	\	用在 **/**（斜杠）、**,**（逗号）或 **=**（等号）之前转义字符；表明下列字符不是元字符。
DNS	.	隔离名称元素。

一些元字符不允许在名称中用作正常字符。可以在名称中使用其它元字符作为正常字符，只要使用反斜杠元字符转义它们即可。

最大名称大小

表35 列出目录服务名称的最大尺寸。注意，限制是特定于实现的，而非特定于体系结构。

表 35. 目录服务名称的最大尺寸

名称类型	最大尺寸（字符）
CDS 简名（两个斜杠之间的字符串）	254
CDS 全名（包括全局或本地前缀、单元名称和隔离简名的斜杠）	1023
DNS 相对名（两个点之间的字符串）	64
DNS 全限定名（所有相对名称的总和）	255

附录B. 对象标识符文件

X/Open 目录服务 (XDS) 接口提供客户应用程序程序员在任何 CDS 中创建和维护名称的能力。程序员还可以创建新的 CDS 属性名称属性类型标签。在 DCE 版本 1.1 目录服务中，每个 CDS 属性名称都具有相应的唯一数字，称为*对象标识符* (OID)。

CDS 提供了一种在对象标识符和人类可读名称之间的转换。这种转换能力使用户能在 DCE 控制程序 (**dcecp**) 接口处输入名称代替输入对象标识符。同时，**dcecp** 在命令输出中显示名称，而不是对象标识符。CDS 属性名和它们相应的标识符存储在称为 **cds_attributes** 的文件中。（请参阅 *IBM DCE 版本 3.2 AIX 和 Solaris 版: 管理指南--简介* 和 *OSF DCE Porting and Testing Guide*， 获取所有 CDS 文件的完整路径名。）

本附录描述了 **cds_attributes** 和 **cds_globalnames** 文件的内容和用法，并说明了应用程序开发者或目录服务管理器如何使用新属性的对象标识符来更新文件。

对象标识符的起始地址

对象标识符的目的是为了确保很多不同的应用程序生成和使用的属性类型中的唯一性。一般，对象标识符是从分配权限层次结构获得的，最高层次是国际标准化组织 (ISO) 和国际电报电话咨询委员会 (CCITT)。个别应用程序开发者不必经常与 ISO 或 CCITT 直接联系以获取唯一的号码。应用程序开发者常从负责分配它们的公司内的人员请求对象标识符。公司权威机构会相应地与更高层权威机构联系，获取唯一的公司前缀。

分配权限的层次结构由分隔对象标识符部分的点表明。点描绘的每个字符串数表示一个分配层次结构级别，从左至右权限依次降低。例如，对象标识符 **1.3.22.1.1.2** 包含下列级别:

1 ISO
3 已标识的组织
22 开放软件基金会
1 分布式计算环境
1 远程过程调用
2 RPC 对象 UUID

cds_attributes 文件

cds_attributes 文件包含 CDS 属性和对象类的对象标识符。下列是缺省文件内容的样本部分:

```
#     OID       LABEL                  SYNTAX
#
1.3.22.1.3.10   CDS_Members            GroupMember
1.3.22.1.3.11   CDS_GroupRevoke        Timeout
1.3.22.1.3.12   CDS_CTS                Timestamp
1.3.22.1.3.13   CDS_UTS                Timestamp
1.3.22.1.3.15   CDS_Class              byte
1.3.22.1.3.16   CDS_ClassVersion       Version
1.3.22.1.3.17   CDS_ObjectUUID         uuid
1.3.22.1.3.19   CDS_Replicas           ReplicaPointer
1.3.22.1.3.20   CDS_AllUpTo            Timestamp
1.3.22.1.3.21   CDS_Convergence        small
1.3.22.1.3.22   CDS_InCHName           small
1.3.22.1.3.23   CDS_ParentPointer      ParentPointer
```

```
1.3.22.1.3.24   CDS_DirectoryVersion    Version
1.3.22.1.3.25   CDS_UpgradeTo           Version
1.3.22.1.3.27   CDS_LinkTarget          FullName
1.3.22.1.3.28   CDS_LinkTimeout         Timeout
1.3.22.1.3.30   CDS_Towers              byte
1.3.22.1.3.32   CDS_CHName              FullName
1.3.22.1.3.34   CDS_CHLastAddress       byte
1.3.22.1.3.36   CDS_CHState         small
1.3.22.1.3.37 CDS_CHDirectories       CHDirectory
1.3.22.1.3.40   CDS_ReplicaState        small
1.3.22.1.3.41   CDS_ReplicaType         small
1.3.22.1.3.42   CDS_LastSkulk           Timestamp
1.3.22.1.3.43   CDS_LastUpdate    Timestamp
1.3.22.1.3.44   CDS_RingPointer         uuid
1.3.22.1.3.45   CDS_Epoch               uuid
1.3.22.1.3.46   CDS_ReplicaVersion      Version
1.3.22.1.3.48   CDS_NSCellname         char
1.3.22.1.3.52   CDS_GDAPointers         gdaPointer
1.3.22.1.3.53   CDS_CellAliases         GroupMember
1.3.22.1.3.54   CDS_ParentCellPointers  ReplicaPointer
1.3.22.1.1.1    RPC_ClassVersion        byte
1.3.22.1.1.2    RPC_ObjectUUIDs         byte
1.3.22.1.1.3    RPC_Group               byte
1.3.22.1.1.4    RPC_Profile             byte
1.3.22.1.1.5    RPC_Codesets            byte
1.3.22.1.5.1    SEC_RepUUID             byte
```

第一列包含 OID，第二列包含标签（标识符映射到的名称），第三列表明数据类型。CDS 数据类型的描述在 **cdsclerk.h** 头文件中。（请参阅 *IBM DCE 版本 3.2 AIX 和 Solaris 版: 管理指南--简介*和 *OSF DCE Porting and Testing Guide*，获取所有 CDS 文件的完整路径名。）

应用程序员不需要修改，除非由于外国语言翻译目的，CDS 标签与 **cds_attributes** 文件中的唯一 OID 关联。但是，程序员可以从适当的分配权限获取新的 OID，为他们自己的对象项创建新的属性，然后将它们附加到现有的列表。

修改文件

当程序员开发使用目录服务的应用程序时，目录服务管理器或应用程序开发者需要为新的应用程序使用的任何新的 CDS 属性名称获取唯一的标识符，然后更新适当的文件。

如果应用程序在 CDS 中存储名称，则编辑 **cds_attributes** 文件。（有关适当的数据类型描述符信息，请参阅 **cdsclerk.h** 文件。）

注: 如果修改 **cds_attributes** 和 **cds_globalnames** 文件中标准属性的 OID 值，可能会遇到与其它目录服务实现交互操作的问题。

修改 CDS 实体属性

每个 CDS 实体都具有属性，它们是与该实体关联的几条或几组数据。属性可以反映或影响实体的可操作行为，记录自上一次启用实体后出现的特定事件或问题的次数，并唯一地区别该实体与任何其它实体。

CDS 属性由 ISO OID 标识。 每个 CDS 属性名称都映射到一个 OID 和一个相应的数据类型。通常，客户应用程序定义属性及其数据类型的名称。应用程序员不需要修改（除非由于外国语言翻译目的）现有的 CDS 标签与 **cds_attributes** 文件中的唯一 OID 关联。但是，程序员可以从适当的权限获取新的 OID，为他们自己的对象项创建新

的属性，然后将它们附加到现有列表中。每个属性的 OID 和数据类型都存储在文件 **cds_attributes** 中。应用程序可以使用的 CDS 数据类型的描述在 **cdsclerk.h** 文件中。

添加新的属性

使用带 **-add** 选项的 **dcecp modify** 操作将新的属性添加到对象实体中。

要添加新的属性，必须预先将新的属性添加到单元中每个主机上的 **cds_attributes** 文件。还必须具有对正在添加新属性的实体的写许可权。

例如，下列命令将单值属性 (**owner**) 添加到目录 (**/.:/admin**)，并将值 **Leland** 赋值到新的属性:

```
dcecp> directory modify /.:/admin -add {owner Leland}
dcecp>
```

下列命令将新的多值属性 (**vegetables**) 添加到对象 (**/.:/admin/garden**)，并将值 **carrots** 和 **lettuce** 赋值到新的属性:

```
dcecp> object
modify  /.:/admin/garden -add {vegetables {carrots} {lettuce}}
dcecp>
```

修改现有属性的值

使用带 **-change** 选项的 **dcecp modify** 操作修改现有属性的值。

要修改属性的值，必须对想修改的属性的名称具有写许可权。

例如，下列命令将 **/.:/admin** 目录的所有者属性值从 **Leland** 更改为 **Peters**:

```
dcecp> directory modify  /.:/admin -change {owner Peters}
dcecp>
```

除去属性

使用带 **-remove** 选项的 **dcecp modify** 操作将属性从对象实体中除去。

要除去一个属性，必须对想除去属性的名称具有写许可权。

要除去一个属性，请使用带 **-remove** 和 **-types** 选项的 **modify** 命令。例如，下列命令将所有者属性从 **/.:/admin** 目录除去:

```
dcecp> directory modify /.:/admin -remove owner -types
dcecp>
```

要从多值属性中除去单一值，请使用 **-remove** 选项并指定要除去的值。例如，下列命令将 vegetables 属性的 carrots 值从 **/.:/admin/garden** 对象中除去:

```
dcecp> object modify  /.:/admin/garden -remove {vegetables
carrots}
dcecp>
```

附录C. 时间提供者和时间服务

本附录说明当选择时间提供者时要使用的标准并描述时间分发服务、时间提供者（硬件和软件）以及它们与 DTS 的交互。本附录还包含世界时区地图。

选择时间源的标准

在为您的网络选择时间源之前，先询问下列问题:

- 提供的时间准确性是多少？

  准确性受时间源本身和传输媒体影响。只要知道不准确性，就可以弥补。

- 时间源的可靠性如何？

  时间源必须可用。如果不可用，则当时间源再次可用时，连接到时间提供者的服务器将使用来自其它服务器的时间来弥补任何时间差异。

- 覆盖的范围是什么？

  时间源必须在时间提供者服务器定位的地理区域中可用。

- 已知不准确性级别是什么？

  如果已知的话，DTS 可以弥补它。大多数源具有已知的不准确性级别。

- 成本是多少？
- 源符合操作环境吗？

  可用的电源和物理条件必须与源兼容；请与供应商联系获取规范信息。

表36为每个时间源类型总结选择标准。

表 36. 时间提供者选择标准

类型	覆盖	不准确性	成本
电话			
NIST	区域的	10 毫秒	每次电话的可变费用
无线电			
MSF	欧洲	10 毫秒	$1K 至 2K
WWV	北美	100 毫秒	$1K 至 2K
WWVB	北美&欧洲	10 毫秒	$1K 至 2K
WWVH	东部&中北部太平洋	100 毫秒	$1K 至 2K
人造卫星			
GOES	世界范围	1 毫秒校正	$2K 至 $20K
GPS	世界范围	< 100 纳秒	$15K 至 $20K

协调世界时间源

存在很多 UTC 时间源，包括电话、无线电和人造卫星。这些将在下列子节中描述。

电话服务

电话时间提供者服务需要时间提供者通过调制解调器拨打中央 UTC 时间源。调制解调器速度和线路的延迟会影响返回时间的准确性。

电话服务通常由标准代理提供。例如，在美国，此项服务是由国家标准技术协会 (NIST) 提供的。除了调制解调器软件成本以外，还需要服务的每次电话费用。

无线电传输

DTS 可以从无线电时间源获取时间。商业接收器监视时间和频率广播，通过时间提供者接口 (TPI) 将时间值返回给 DTS 服务器。 NIST 操作下列美国时间和频率站:

- WWV

  以 2.5, 5.0, 10.0, 15.0 MHz 发送到北美和南美。
- WWVB

  倘若是高质量频率信息，则主要以 60 kHz 发送到美国，因为大气传播影响相对较小。
- WWVH

  以 2.5, 5.0, 10.0, 15.0 MHz 发送到阿拉斯加、夏威夷、澳大利亚、新西兰、日本和东南亚。

下列站在欧洲可用:

- MSF

  以 60 kHz 从英国广播。
- DCF77

  以 77.5 kHz 从德国广播。

除了上面列出的站以外，在世界范围内还有 30 个无线电站提供 UTC 时间。详情请与您的国家中的国家标准组织商议。

网络时间协议

具有因特网访问权的节点可以使用网络时间协议 (NTP) 作为 DTS 的 UTC 时间源。(有关如何将 NTP 用作时间提供者的解释，请参阅第243页的『第26章 与“网络时间协议”的互操作』。)

人造卫星

人造卫星具有世界范围的可用性；如果它们的延迟可知且能弥补的话，它们能提供相对精确的时间。有关 UTC 的人造卫星源，请参阅下列列表:

- GOES

  与地球的相对位置不变的可操作环境人造卫星
- TRANSIT

  美国海军人造卫星系统，包括四个跟踪系统和两个地人造卫星通信站点
- GPS

  全局定位系统，人造卫星接收器

世界时区地图

图79 显示了世界时区地图，包括下列内容:

- UTC 引用区
- 奇数区和偶数区
- 半时区
- 未采用区系统或时间与相邻区相差不为半个小时的国家和区域

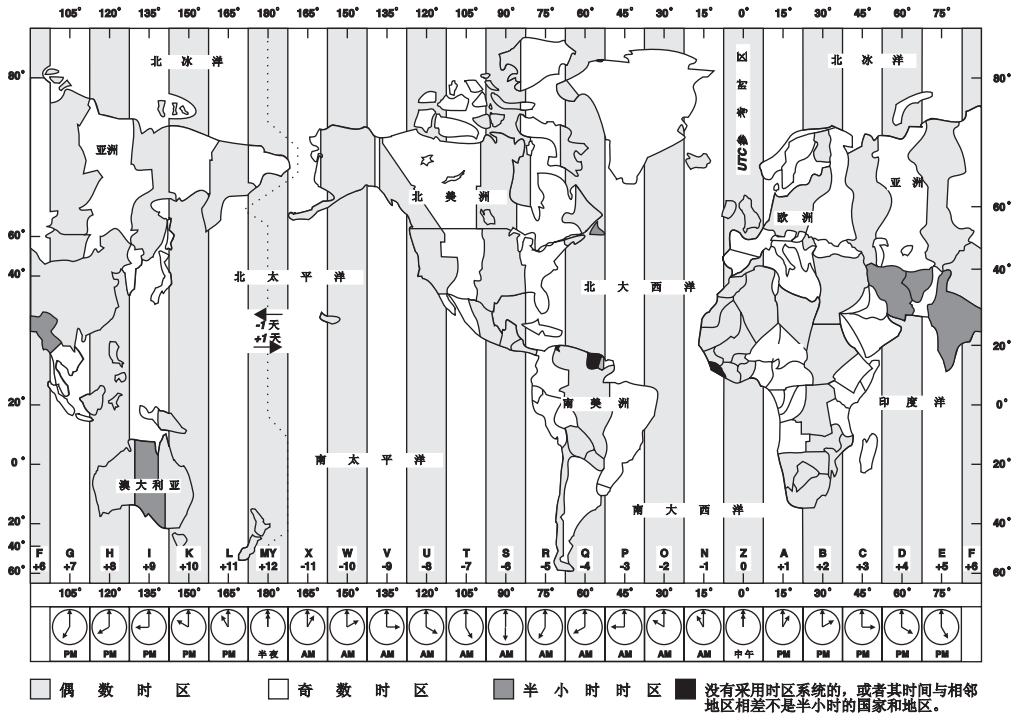


图 79. 世界时区地图

附录D. DTS 扩展的 BNF

本附录以扩展 Backus Naur 格式 (BNF) 记数法定义分布式时间服务 (DTS) 语法。

DTS 时间转换的 BNF 具有四个部分: *年*、*日*、 *tdf* 和*不准确性*。对于值未明白表示的任何部分，转换缺省值将采用当前日。 DTS 时间转换的 BNF 如下所示:

```
dts_time :  year_part day_part tdf_part inacc_part
|   year_part day_part tdf_part
|   year_part day_part
|   year_part day_part inacc_part
|   year_part inacc_part
|   year_part
|   day_part  tdf_part inacc_part
|   day_part  tdf_part
|   day_part inacc_part
|   day_part
|   year_part Z
|   year_part Z inacc_part
|   year_part day_part Z inacc_part
|   day_part  Z inacc_part
|   day_part  Z
;
year_part : number - number - number -
| number - number - number T
| number - number T
| number T
;
day_part  : partial : partial : partial
| partial : partial
| partial
;
tdf_part  : sign number : number
| sign number
;
sign      : -
| +
;
partial   : number
| number frac
| number frac number
| frac number
;
frac      : .
| ,
;
inacc_part : I
| I partial
| I infinity
;
infinity   :  'i''n''f'
|  - -
|  - - - - -
;
number     :  DIGIT
|  number DIGIT
;
```

附录E. 声明

本信息针对的是在美国提供的产品和服务。IBM 可能未在其它国家提供本文档中讨论的产品、服务或特性。有关本区域当前可提供的产品和服务信息，请询问当地 IBM 代表。任何对 IBM 产品、程序或服务的引用并不说明或暗示只能使用 IBM 的产品、程序或服务。任何不侵犯 IBM 知识产权且具有同等功能的产品、程序或服务都可以用来代替 IBM 产品、程序或服务。但是，对任何非 IBM 产品、程序或服务运作的评估和验证由用户自行负责。

IBM 可能已经申请或正在申请与本文档中主题有关的各项专利。提供本文档并不表示允许使用这些专利。您可以用书面方式将许可查询寄往:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

有关双字节 (DBCS) 信息的许可证查询，请与本国的 IBM 知识产权部门联系，或将书面查询寄往:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

以下段落不适用于英国或条例与本地法律不一致的任何其它国家: 国际商用机器公司以 『″AS IS″』 方式提供本出版物，但不提供任何类型的担保，无论是明确的还是隐含的，其中包括但不局限于对于特定目的的合法性、适销性或适用性隐含担保。一些国家不允许在特定交易中拒绝明确的或隐式的担保，因此本声明可能不适用。

本信息可能有技术上的不准确性或者印刷错误。周期性的修改将写入信息；这些更改也将加在新版的信息中。IBM 可能随时对本信息中描述的产品和（或）程序做改进并（或）更改，请恕不再另行通知。

在本信息中任何引用非 IBM 的 Web 站点仅为方便起见，并不代表对这些 Web 站点的认可。那些 Web 站点的资料不是本 IBM 产品资料的一部分，使用那些 Web 站点的风险自负。

IBM 可能会以它认为适当的任何方式使用或散发您所提供的信息，而不必对您负任何责任。

为以下目的: (i) 允许在独立创建的程序和其它程序（包括本程序）之间进行信息交换 (ii) 允许对已经交换的信息进行相互使用，而希望获取本程序有关信息的合法用户请与下列地址联系:

IBM Corporation
Department LZKS
11400 Burnet Road
Austin, TX 78758
U.S.A.

依照适当的条款和条件，其中包括在一些情况下需要付费，这些信息或许是可用的。

本文档中描述的特许程序及其所有可用的特许资料都由 IBM 根据 IBM 用户条例、 IBM 国际程序设计许可证条例或任何等价的条例来提供。

这里包含的任何性能数据都是在受控的环境中确定的。因此，在其它操作环境中获得的结果可能有明显的不同。可能已经在开发级别的系统上进行了一些测试，不能保证这些测试在通常可用的系统上结果相同。此外某些测试可能是通过推断估计得出的。实际的结果可能不同。本文档的用户应该验证针对他们特定环境的适用数据。

非 IBM 产品的信息请从其供应商、其发布的公告或其它公共资料来源处获得。IBM 从未测试过这些产品，并无法确认与非 IBM 产品相关的性能准确性、兼容性或任何其它索赔。有关非 IBM 产品的功能应从其供应商处获得。

将更改或撤销所有关于 IBM 未来发展方向或意图的声明而只提出目的和目标，请恕不另行通知。

显示的所有 IBM 价格是它暗示的当前零售价，可能随时改变，请恕不另行通知。经销商的价格可能不同。

该信息包含日常商业操作中使用的数据和报告。为了尽可能完整的阐明它们，示例包含个人、公司、商标和产品的名称。所有这些名称均为虚构，如果它们与任何实际商业企业的名称和地址类似纯属巧合。

版权许可:

该信息包含用源语言编写的样本应用程序，它阐明在各种操作平台上的编程技巧。为开发、使用、销售或分发与编写样本程序操作平台的应用程序编程接口一致的应用程序的目的，您可以以任何形式复制、修改和分发这些样本程序，而不必向 IBM 付费。

这些样本未在所有条件下充分测试。因此 IBM 无法保证或暗示这些程序的可靠性、适用性与功能。为开发、销售或分发与应用程序编程接口一致的应用程序，您可以不向 IBM 付费而以任何形式复制、修改和分发这些样本程序。

这些样本程序的每个副本或任何部分或任何派生工作必须包含如下版权注意事项:

© （您公司的名称）（年份）。本代码的一部分从 IBM 公司样本程序获取。© Copyright IBM Corp. *1990, 2001*. All rights reserved.

如果您正在查看该信息的软拷贝，可能不会出现照片和彩色插图。

商标

下列术语是“国际商用机器公司”在美国或（和）其它国家的商标:

- AIX
- DFS
- Encina
- IBM
- MVS

UNIX 是 Open Group 在美国和其它国家的注册商标。

Microsoft、Windows、Windows NT 和 Windows 徽标是 Microsoft 公司在美国或（和）其它国家的注册商标。

其它公司、产品和服务名可能是其它公司的商标或服务标志。

索引

[A]

[B]

[C]

[H]

[J]

[K]

[L]

[M]

[P]

[Q]

[R]

[S]

[T]

[W]

[X]

[Y]

[Z]

A

B

C

D

E

F

G

I

K

L

M

N

O

P

Q

R

S

T

U

W

X

[特别字符]

Printed in China